Single sign-on i federació d'identitats

José María Palazón Romero
Antoni Felguera
Jordi Castellà-Roca

PID_00183942

UOC
Universitat Oberta de Catalunya

Els textos i imatges publicats en aquesta obra estan subjectes –llevat que s'indiqui el contrari– a una llicència de Reconeixement-NoComercial-SenseObraDerivada (BY-NC-ND) v.3.0 Espanya de Creative Commons. Podeu copiar-los, distribuir-los i transmetre'ls públicament sempre que en citeu l'autor i la font (FUOC. Fundació per a la Universitat Oberta de Catalunya), no en feu un ús comercial i no en feu obra derivada. La llicència completa es pot consultar a http://creativecommons.org/licenses/by-nc-nd/3.0/es/legalcode.ca

Índex

Introducció

Des del començament de la informàtica, els diferents sistemes han necessitat la capacitat de limitar l'accés als recursos computacionals disponibles. Segons el cas, les motivacions han oscil·lat des de la vigilància dels enormes costos d'aquests recursos en els anys seixanta fins avui dia, en què per motius de seguretat el control d'accés a programes o dades emmagatzemades es deixa entreveure com un requisit imprescindible.

De manera genèrica, la seguretat de la informació té tres dimensions essencials: la disponibilitat, la integritat i la confidencialitat. Mitjançant la disponibilitat volem expressar la necessitat que els sistemes estiguin disponibles quan siguin requerits, i garantir així la prestació de les funcions per les quals han estat dissenyats.

D'altra banda, la integritat es refereix a la propietat que ningú, excepte els usuaris permesos, no pot alterar la informació d'una altra manera que no sigui la permesa per les polítiques de seguretat. Finalment, la confidencialitat enuncia que ningú, excepte els usuaris permesos, no pot saber el contingut de la informació protegida.

Aquesta limitació i aquest control d'accés d'un subjecte als recursos ofereix concretament característiques d'integritat i confidencialitat a la protecció de la informació. Amb aquesta limitació es pretén evitar que els subjectes no permesos alterin la informació protegida, i també que accedeixin al contingut d'aquesta informació i la recuperin. Amb aquest objectiu, s'han definit diferents operacions, entre les quals trobem la identificació del subjecte, la seva autenticació o el control d'accés.

Abans, quan els sistemes eren monolítics i no tenien connexió entre si, les operacions esmentades eren gestionades íntegrament pel sistema al qual es volia accedir. Del conjunt abstracte format pels usuaris, recursos, permisos i registres d'accés del sistema vinculats a la identificació i el control d'accés, se n'ha dit *domini d'identitat* o *seguretat*.

A mesura que els sistemes d'informació es connectaven entre si, va sorgir la necessitat que hi hagués diferents subjectes de diferents dominis d'identitat que poguessin accedir a recursos d'altres dominis en els quals es confiava. És en aquest moment quan sorgeix la idea de la federació d'identitats, basada en el concepte d'integrar i coordinar diferents dominis d'identitat entre si amb l'objectiu de formar un domini més gran per a la gestió i el control de l'accés, fruit de la suma de les característiques individuals que tenien. La federació d'identitats permet donar serveis complexos, però no és menys cert que la seva evolució, com gairebé tot el que està relacionat amb la identitat digital, va

néixer i va créixer de manera fragmentada, amb múltiples iniciatives atomitzades que s'han anat desenvolupant de manera més o menys paral·lela, a la llum dels esforços de les grans multinacionals i agrupacions creades.

Addicionalment, també cal destacar que la federació d'identitats no s'usa regularment a l'hora de construir sistemes d'informació, la qual cosa dificulta més si fos possible que el gran públic l'adopti. Únicament acrònims com el de *single sign-on* (SSO) són més o menys coneguts com una *commodity* de mercat, però lluny de constituir un veritable requisit a l'hora de planificar la construcció d'un sistema d'informació.

En una societat enormement connectada, amb serveis dinàmicament compostos, una alta fluctuació de les relacions i una estreta relació entre el món físic i el virtual, la federació d'identitats està cridada a ser, no una opció, sinó una necessitat apressant de desenvolupament de l'anomenada *Internet del futur*.

Objectius

En aquest mòdul l'estudiant trobarà els continguts i les eines que li permetran assolir els objectius següents:

1. Comprendre el concepte de *federació d'identitats*, els termes associats i la seva relació amb els diferents elements de la gestió de la identitat i l'accés.

2. Adquirir el coneixement dels diferents patrons de federació d'identitat que hi ha basats en l'establiment de la confiança inicial entre els parells.

3. Conèixer els diferents estàndards de federació d'identitat, a més de les tecnologies associades i els seus principals components i accions.

4. Entendre el funcionament dels mètodes i de les tecnologies que permeten la federació d'identitats encara que no hagin estat dissenyats estrictament per a això.

Tots aquests objectius els durem a terme amb una exposició senzilla i clara de la matèria, conduïda amb uns exemples pràctics introduïts al començament.

Mostrarem, a més, porcions de codi de programació que il·lustraran més concretament els fets específics dels protocols més importants. És rellevant destacar que amb aquests exemples no pretenem convertir aquest mòdul en un manual de programació d'aquests protocols, sinó simplement introduir l'estudiant en les operacions més importants i la morfologia concreta a baix nivell d'aquestes operacions.

1. La federació d'identitats

En aquest apartat, i a partir d'exemples pràctics senzills, s'introdueixen els conceptes previs necessaris per a definir la federació d'identitats, el *single sign-on*, i també els diferents patrons de federació que hi ha.

En els dos exemples següents es presenta la federació d'identitats com una solució a problemes reals que hi ha, i serviran per a explicar de manera didàctica els conceptes del mòdul.

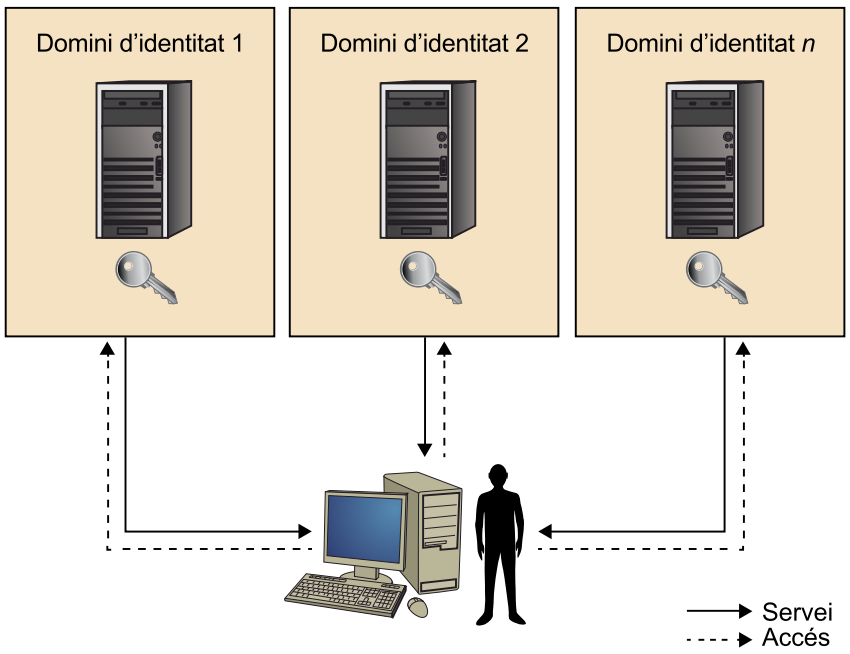
Unes vacances còmodes

Quan ens plantegem fer unes vacances, no és estrany que el primer que vulguem assegurar siguin els vols disponibles a la destinació que hem triat. És fàcil que accedim al lloc web de la companyia aèria seleccionada, reservem el vol i ens demanin diligentment si ja hi havíem fet un registre. Si l'hem emplenat en una altra ocasió, hi solem introduir el nostre usuari i la nostra contrasenya perquè el proveïdor recuperi les nostres dades. Si no, hi introduïm llavors totes les nostres dades, tant personals com econòmiques, perquè la companyia aèria pugui completar la transacció. Finalment, confirmem totes les dades de l'operació i pagem.

A continuació ens adonem que necessitem un cotxe per a anar, per exemple, al nostre hotel. Per a llogar-ne un, triem la nostra companyia de lloguer de cotxes favorita, seleccionem el producte que volem i tornem a fer una altra vegada tot el procés de registre o recuperació de les nostres dades, la confirmació i l'execució de l'operació.

Si aquest procés es produeix diverses vegades perquè, per exemple, havíem de contractar també un hotel a la zona, reservar entrades per a unes atraccions, transports especials, etc., tornem a fer l'operació cada vegada. De tot plegat en surt un esquema com el que es mostra en la figura:

Accés a molts proveïdors de servei aïllats amb moltes credencials



Com es mostra en l'esquema, l'usuari té tantes identitats com proveïdors de servei hi ha, quan en realitat és la mateixa persona. Els proveïdors de servei es mantenen aïllats entre si, de manera que no integren les seves propostes ni tampoc la seva gestió d'identitats.

L'usuari, per la seva banda, ha de recordar tants usuaris i contrasenyes com proveïdors de servei hi ha i ha de proporcionar la mateixa informació cada vegada a tots i cadascun dels proveïdors de servei amb els quals vol tenir tractes. Finalment, quan l'usuari modifiqui les dades personals o les vulgui esborrar, al seu torn, les haurà de canviar o esborrar en tots i cadascun dels proveïdors de servei que hi ha.

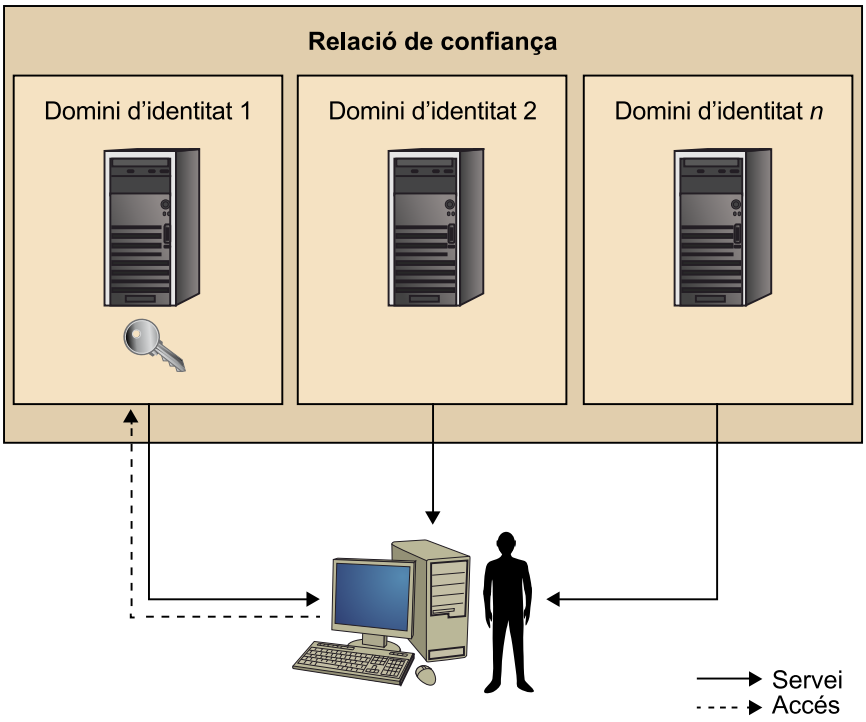
El servei mundial de mobilitat acadèmica

La comunitat acadèmica s'ha caracteritzat sempre per una gran mobilitat física. Hi ha molts professors que viatgen contínuament d'una universitat a una altra per fer-hi estades o impartir classes. Els estudiants també es beneficien de facilitats per a la mobilitat, com les beques Erasmus, que permeten l'intercanvi d'estudiants europeus entre universitats del continent.

Així, fruit d'aquesta necessitat es va crear EduRoam, un servei de mobilitat segura per a la comunitat acadèmica mundial. Amb aquest servei qualsevol membre de la comunitat acadèmica que pertanyi a una institució que hi estigui vinculada pot viatjar a qualsevol altra institució i, immediatament, tenir accés a recursos, com per exemple la connexió a Internet.

Com es veu en la figura, els usuaris (professors, administratius, estudiants, etc.) no han de demanar a la institució de destinació que els faciliti credencials, ja que en tenen unes en la seva organització d'origen, amb la qual tenen una relació de confiança que els permet autenticar les credencials mostrades i donar accés immediat a una sèrie de recursos, com la connexió a Internet.

Accés a molts proveïdors de servei amb un únic joc de credencials



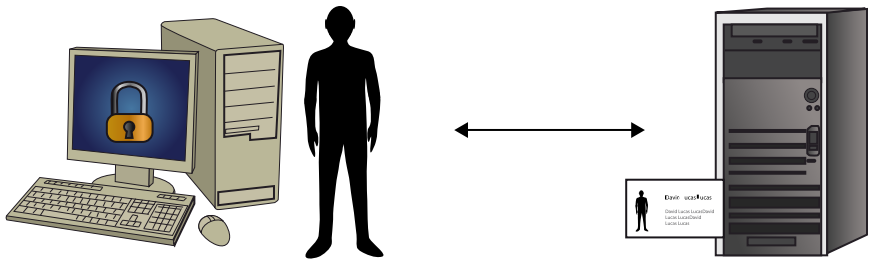
1.1. Conceptes previs per a la definició de la federació d'identitats

En aquest subapartat presentarem els conceptes previs essencials de la federació d'identitats abans de fer-ne la definició. Aquests conceptes previs ens permeten reforçar i aclarir idees que utilitzarem més endavant de manera repetida.

1.1.1. Començar el camí

Encara que parlem de federació d'identitats, el concepte bàsic que es continua mantenint, l'element essencial, és el d'un subjecte que vol accedir a un objecte de manera segura, tal com es mostra en la figura:

Esquema conceptual base per a serveis d'identitat



Aquest primer acostament és necessàriament molt genèric. Encara que sembli que el **subjecte** és simplement una persona i que l'**objecte** són les dades, la veritat és que l'evolució de la gestió d'identitats i l'aplicació d'aquesta gestió en els diferents paradigmes actuals i futurs de la computació ha generalitzat aquests conceptes. De manera no exclusiva, diem que:

- Un **subjecte** pot ser una persona però també un *thread*, o un dispositiu, per exemple.

- Un **objecte** són dades, però també serveis, recursos computacionals o qualsevol bé que custodiï un sistema informàtic (físic o lògic).

- L'expressió *de manera segura* pretén assegurar que es proveeixen les dues característiques bàsiques per les quals està configurat: integritat i confidencialitat.

Per a complir aquest requisit, la federació d'identitats es nodreix de les operacions bàsiques de la gestió d'identitats:

- **Identificació**. Hi ha pocs sistemes d'identitat que ofereixin la possibilitat d'identificar el subjecte de manera unívoca. Entre aquests sistemes, n'hi ha alguns de biomètrics avançats, sempre que l'univers d'individus no sigui massa gran.

- **Autenticació**. És un procés pel qual el sistema constata, mitjançant algun mecanisme, que l'usuari és qui diu que és. Hi ha molts mètodes diferents d'autenticació, basats essencialment en tres aspectes: el que se sap, el que es té o el que s'és.

Sistemes biomètrics d'identificació

Els sistemes biomètrics d'identificació es basen en les característiques físiques o de comportament de les persones. Aquestes característiques poden ser el reconeixement d'empremta dactilar, de veu, de retina, la manera de caminar, etc.

- **Autorització** o **control d'accés**. És un mecanisme amb el qual es permeten les operacions d'accés o alteració dels béns que s'estan protegint.

Exemple d'autorització

Un exemple clar d'autorització és el control que exerceix el sistema operatiu en intentar accedir als nostres fitxers, basant-se en la configuració concreta dels permisos de l'usuari.

- **Registre**. Quan accedim a un determinat bé, una vegada estem identificats i autenticats i hem superat el control d'accés, hi ha una altra operació bàsica anomenada *registre*, mitjançant la qual es guarden totes les operacions que s'han executat sobre un determinat recurs, a fi de tenir un quadern de bitàcola de les accions que s'han dut a terme.

Dels protocols que tenen aquestes tres últimes operacions se'n diu *protocols AAA* (*authentication*, *authorization*, *accountability*). Els sistemes d'identitat federats, com que són sistemes d'identitat en si mateixos, han de tenir les mateixes operacions que un sistema d'identitat en ús. L'usuari en disposarà perquè les implementa el sistema específicament, o bé perquè les hereta d'un dels sistemes federats.

1.1.2. La dificultat de federar identitats

Com hem vist en els exemples al començament de l'apartat, quan dues entitats decideixen federar els seus sistemes d'identitat i control d'accés, han de garantir les operacions bàsiques i les característiques de seguretat que han de proveir. És a dir, quan la companyia aèria decideix federar els seus sistemes amb la companyia de lloguer de cotxes, han de proveir conjuntament les capacitats d'autenticació i registre o la de control d'accés als recursos, que doten aquests usuaris de les característiques d'integritat i confidencialitat que hem esmentat més amunt. Pensem, a més, que cadascun dels sistemes pot haver estat ideat de manera diferent, amb diferents estructures, sistemes de privilegis o conceptes base, de manera que posar-los a funcionar junts no és, en principi, una cosa òbvia.

Com a mostra d'aquesta complexitat, observem que únicament per al control d'accés, per exemple, el món de la gestió de la identitat oscil·la entre els sistemes DAC[1], basats en una llista d'accés per recurs, els sistemes RBAC[2], basats en rols, i els sistemes MAC[3], per a sistemes militars, entre d'altres. La fragmentació en altres operacions, com la d'autenticació, és encara molt més gran, amb molts mètodes i moltes tècniques amb diferents nivells de seguretat.

(1)*DAC* és la sigla de *control d'accés discrecional* o *discretionary access control*.

(2)*RBAC* és la sigla de *control d'accés basat en rols* o *role-based access control*.

(3)*MAC* és la sigla de *control d'accés obligatori* o *mandatory access control*.

Aquesta contínua proliferació d'acostaments a l'hora d'escometre la implementació dels sistemes de gestió d'accés va plantejar dificultats a l'hora d'acostar-se a les possibles solucions respecte a la federació d'identitats. El repte era sobre la taula: si tots els sistemes són tan diferents els uns dels altres, com aconseguirem comunicar-los entre si?

1.1.3. Propostes inicials

Una vegada plantejat el repte, es van proposar diferents solucions abans d'arribar fins al que avui coneixem com a **sistemes d'identitat federats**. Aquestes solucions estaven basades en una centralització d'algun dels elements clau que tenien, que permetia una coordinació posterior entre els diferents sistemes que el componien. No podem anomenar aquests sistemes *federats*, ja que realment no constituïen una federació a causa de la relació jeràrquica que tenien. Concretament, podem classificar les propostes de la manera següent:

- **Metadirectoris**. És un repositori d'informació amb la còpia de la informació de tots els repositoris d'identitat. Se sincronitza cada cert temps per a mantenir l'estat dels diferents repositoris.

- **Directoris virtuals**. De la mateixa manera que els metadirectoris, són repositoris d'informació, encara que en lloc de tenir una còpia de la informació dels diferents repositoris tenen apuntadors en els quals hi ha realment la informació.

S'ha escrit molt sobre l'adequació dels sistemes centralitzats a l'hora de resoldre els problemes que hem exposat. Les crítiques principals consisteixen en l'escalabilitat de la solució, ja que mantenir identificadors únics (ID canònics únics) pot ser una tasca molt complexa a causa del gran nombre d'actors que, potencialment, hi pot haver involucrats.

Una altra de les crítiques és l'adaptació d'aquests sistemes a la realitat específica d'organitzacions i aplicacions, és a dir, que el manteniment d'ID canònics únics pot afectar no solament el manteniment (adaptació enfront de canvis) i l'escalabilitat d'aquests sistemes, com hem vist en el paràgraf anterior, sinó també la capacitat d'adaptació a diferents realitats. Un entorn centralitzat implica una sèrie de característiques a l'hora d'autenticar, de mantenir un conjunt de privilegis amb una determinada estructura, etc., que pot ser que no sigui adequada a diferents realitats. Per exemple, un sistema RBAC, basat en rols i molt estès en ambients empresarials, pot ser que no sigui adequat en un entorn militar, en què les prioritats són unes altres i estan molt estesos els sistemes MAC. Per això es fa difícil pensar que un entorn centralitzat pugui satisfer tots els requisits de les diferents realitats organitzacionals que hi pot haver.

Encara que un entorn centralitzat constitueix un únic punt de control, cosa que permet una focalització d'esforços a l'hora d'assegurar i administrar un sistema d'informació, també constitueix un únic punt de fallada, cosa que augmenta el risc de no-disponibilitat al conjunt dels sistemes a què dóna servei.

Totes aquestes raons, això és, l'escalabilitat, l'adaptació a les diferents realitats i l'únic punt de fallada, han constituït a la pràctica barreres insalvables per a aquest tipus de sistemes, tot i que en determinats entorns (per exemple, dins d'una única empresa amb polítiques homogènies) encara continuen vigents. La complexitat de la federació d'identitats ha ultrapassat les capacitats d'aquest tipus d'entorns i ha donat pas a diferents acostaments fruit de diferents gènesis, que convergeixen de mica en mica i que mantenen com a denominador comú el respecte per la gestió distribuïda dels recursos i, per tant, dels sistemes d'identitat i accés.

1.2. Definició de *federació d'identitats*

És important, abans de continuar endavant, que definim els diferents conceptes que utilitzarem repetidament al llarg del mòdul a fi d'aclarir-los i puntualitzar-los abans que condueixin a qualsevol mena d'equivocació.

Segons hem vist més amunt, la federació d'identitats manté els mateixos objectius que un sistema d'identificació i gestió d'accés en ús, és a dir, el d'un subjecte que vol accedir a un objecte de manera segura. Per això ja hem definit en el subapartat 1.1 els conceptes de *subjecte*, *objecte* i *de manera segura*.

Ara estem en disposició de formalitzar les definicions que intervindran en un sistema d'identitat federat, tal com es mostra en la figura:

- Un **subjecte** és qualsevol element capaç de presentar-se davant un sistema d'identificació per a reclamar una identitat i autenticar-se davant aquest sistema.

- Un **objecte** és un bé accessible per un subjecte servint-se de mitjans electrònics.

- Una **taula de capacitats** són les capacitats d'accés que té un determinat subjecte als diferents objectes protegits per un sistema.

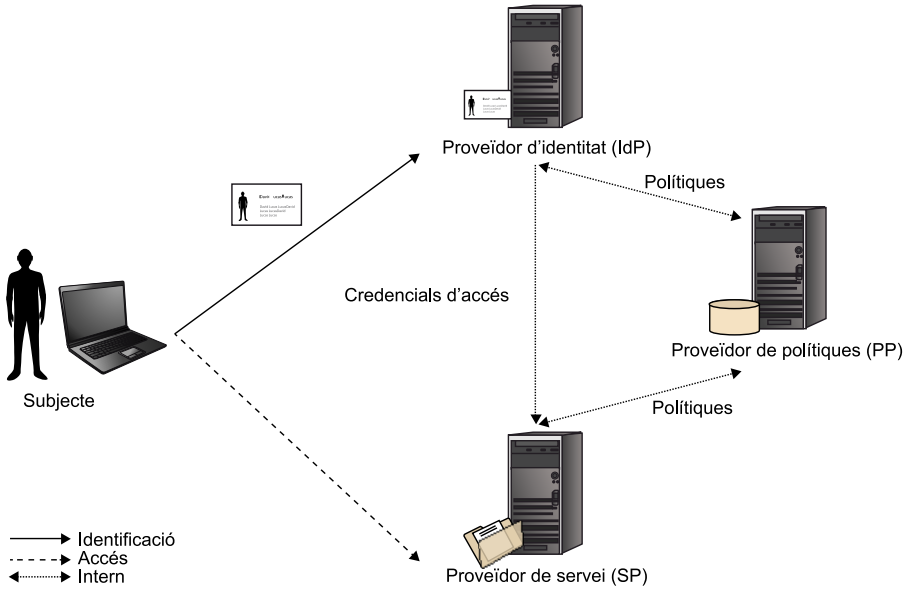
- Un **domini d'identitat** (o **domini de seguretat**) és un conjunt de subjectes, objectes i mecanismes per a la identificació i la gestió d'accés.

- Un **proveïdor de servei** (SP) és un sistema que proveeix dels objectes del sistema i els posa a la disposició dels subjectes pertinents.

- Un **proveïdor d'identitat** (IdP) és un sistema que proveeix dels mecanismes propis d'un sistema d'identificació i control d'accés.

- Un **proveïdor de polítiques** (PP) és un sistema que configura, emmagatzema i distribueix les polítiques de seguretat referides a la identitat i al control d'accés.

Diagrama de relació entre els diferents actors d'un sistema d'identitat



Tal com veiem en la figura anterior, entre el subjecte i el proveïdor d'identitat es duu a terme el procés d'identificació i autenticació mitjançant una aportació de credencials per part del subjecte. En aquest procés també intervé el proveïdor de polítiques, oferint les polítiques d'identitat necessàries, com per exemple les restriccions horàries per a l'autenticació. En aquest moment, el proveïdor d'identitat certifica la identitat d'un subjecte al proveïdor de servei, que, d'acord amb la taula de capacitats d'aquest subjecte, permetrà l'operació requerida o no.

La **federació d'identitat** es pot definir com els mecanismes necessaris per a permetre a una identitat pertanyent a un domini d'identitat concret operar amb objectes d'un proveïdor de servei associat a un domini d'identitat federat, d'acord amb les polítiques de seguretat establertes per a cadascun d'aquests objectes i per a la federació en conjunt.

Nota

En aquest moment s'investiguen sistemes de reputació i confiança que ajudin a establir relacions de federació d'identitats sense coneixement previ, però encara estan en estadis molt inicials.

Els sistemes d'identitat federats pretenen, per dir-ho així, reunir les diferents identitats que hi ha d'un determinat subjecte en molts dominis d'identitat. Per això, cal assumir que, perquè hi hagi una federació d'identitats, necessitem que hi hagi més d'un domini d'identitat. Si només hi hagués un domini d'identitat i els recursos confiessin en la gestió que fa aquest domini sobre la identificació i el control d'accés, no parlaríem de federació d'identitats sinó de control d'accés delegat, de la qual cosa també tractarem més endavant a manera d'aclariment.

És important destacar que els sistemes d'identitat federats constitueixen, en si mateixos, una relació de confiança entre dues parts. Aquesta confiança ha d'estar definida prèviament perquè la tecnologia la pugui comunicar després, per exemple, mitjançant diferents tecnologies com SAML. En tot cas, no és possible tecnològicament establir *a priori* la confiança inicial sense aquesta relació prèvia esmentada fora de l'ambient tecnològic.

Vegeu també

Les tecnologies SAML es descriuen en el subapartat 3.1 d'aquest mòdul.

1.3. Patrons de federació

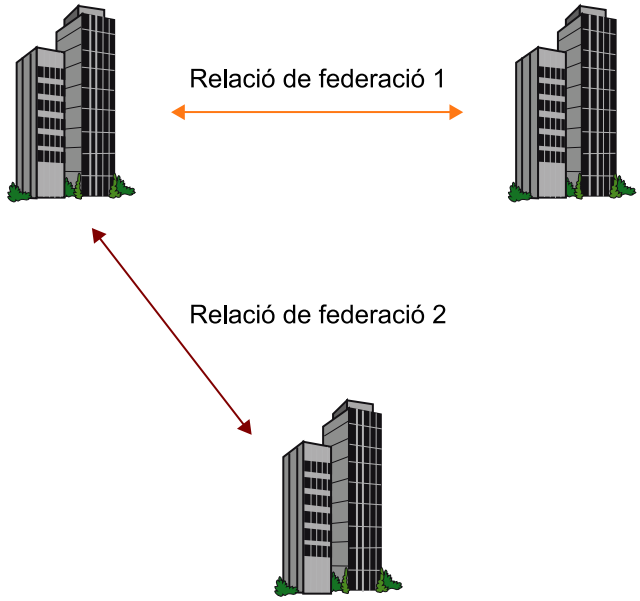
Un sistema d'identitat i gestió d'accés implica diferents aspectes a part del tecnològic. Quan pensem en això no solament hem de tenir en compte la manera d'autenticar-se com a element més evident i exposat, sinó també, per exemple, la forma de distribució de les credencials, la política d'ús, el registre que durà a terme, les evidències electròniques, la privadesa, etc.

Com a exemple d'això últim, en la federació d'identitats és molt important com s'estableix el nivell de confiança inicial entre dos dominis d'identitat com a part integral del procés de desplegament d'aquest acord de federació. Depenent d'això s'estableixen diferents patrons de federació, que impacten decididament en els resultats del procés de federació.

1.3.1. Federació *ad hoc*

La federació *ad hoc* és un tipus de patró de federació en el qual la relació es fa 1 a 1 entre parelles, tal com es veu en la figura. Tots els acords relatius es fan entre dos dominis d'identitats diferents, i potencialment pertanyents a dues organitzacions també diferents. A causa d'això, si hi ha una altra relació *ad hoc* dins d'una mateixa empresa, s'hauran de tornar a negociar exactament igual tots els termes de l'acord de federació d'identitats amb la destinatària d'aquesta altra relació de federació.

Diagrama conceptual de patrons de federació *ad hoc*



Generalment, no es disposa d'estàndards i, si se'n tenen, han de ser adoptats de la mateixa manera per les dues parts de la relació. En aquest sentit, les decisions de la federació tenen un alt impacte en la manera com les identitats es gestionen en cadascuna de les parts, cosa que implica que realment no es fa gaire per a solucionar els problemes dels acostaments centralitzats exposats més amunt.

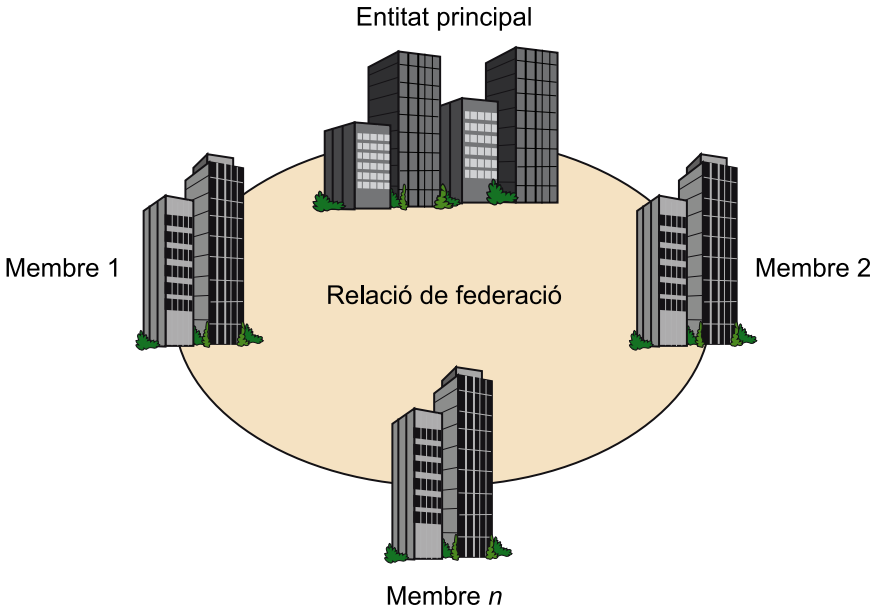
D'aquest tipus de relacions n'hi ha des de fa temps, generalment com a resposta a les oportunitats de negoci temporals o bé a les relacions de dependència d'algunes empreses respecte d'altres de més grans, com és el cas de les empreses subministradores de grans conglomerats empresarials.

En aquest cas, el fet de dur a terme una relació de federació no ha de ser gaire traumàtic però la complexitat varia depenent del nombre de relacions que cal adoptar, ja que la quantitat de treball que comporta és lineal. Així, la capacitat d'una empresa de mantenir diferents relacions de federació *ad hoc* depèn de la seva capacitat de gestionar moltes relacions independents i els costos tècnics i econòmics associats a aquestes relacions.

1.3.2. Federació *hub and spoke*

La federació *hub and spoke* és un tipus de federació que es basa en una organització dominant que dicta el conjunt d'estàndards tècnics, les polítiques de confiança, la gestió del risc, etc., i un conjunt d'organitzacions, generalment de menys grandària, que segueixen aquests dictàmens, de manera que es crea un arxipèlag pel que fa a la identitat.

Diagrama conceptual de federació *hub and spoke*



Els acords són 1 a 1 i sempre hi ha una de les entitats que és la principal del concentrador o *hub*, cosa que no facilita l'escalabilitat del patró.

Exemple

Reprenent el primer exemple descrit al començament de l'apartat, podríem pensar que la companyia aèria en si mateixa pot ser l'entitat principal per la força econòmica que té, i que diferents empreses més petites, com les de lloguer de cotxes o els serveis d'hotel, poden ser membres de la federació. En la realitat, hi ha grans empreses com Boeing, Amazon o General Motors que tenen aquest tipus de patró de federació.

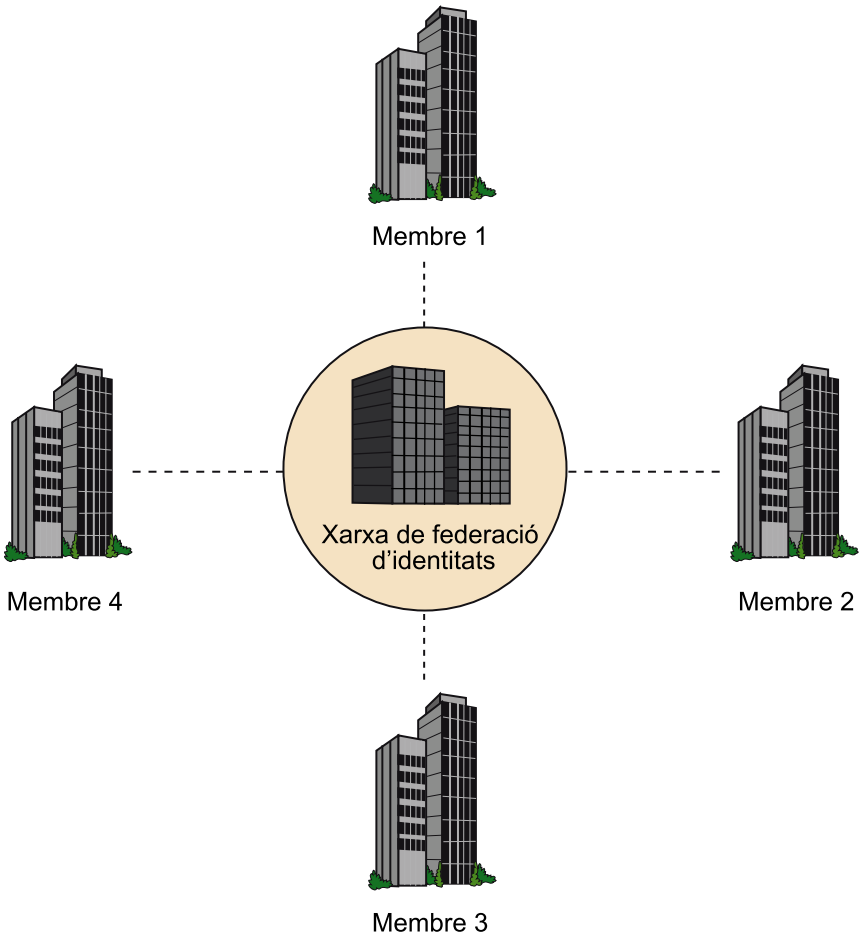
Si alguna de les organitzacions membres de la federació s'ha de federar amb una altra organització que també té un patró del mateix estil, possiblement haurà de tornar a fer els mateixos passos que va seguir per unir-se a la primera federació, ja que la inversió feta en la primera no li serviria de gaire, perquè segurament hauria d'adoptar diferents estàndards tècnics, legals o de qualitat de servei.

Finalment, s'ha de recalcar que la confiança de la federació l'aporta gairebé exclusivament l'empresa de més grandària, que és la que dicta les normes i les polítiques que regulen la federació. En un patró d'aquest tipus, és freqüent observar que si hi ha empreses de grandària similar que federen identitats, cadascuna d'elles tendirà a protegir els seus interessos, cosa que donarà lloc moltes vegades a conflictes interns.

1.3.3. Xarxa de federació d'identitats

Una xarxa d'identitats és una organització independent completa i únicament dedicada als aspectes tècnics i administratius de la federació d'identitats. Hi és exclusivament perquè la xarxa funcioni de manera adequada i serveixi als seus membres, que són realment els propietaris de la xarxa en si mateixa. Els membres grans i petits tenen tots els mateixos deures i obligacions, fins i tot si competeixen entre ells.

Diagrama conceptual de federació mitjançant una xarxa de federacions



Amb aquesta organització, la xarxa es pot centrar a dotar de valor la mateixa xarxa gestionant, per exemple, els aspectes relatius a la privadesa, o la relació amb els governs i els ens legals, o bé lluitant contra el frau i el robatori d'identitat. La xarxa estipula també una sèrie de normes comunes i té articulats mecanismes de resolució de conflictes per a la no-obediència de les normes establertes, que pot arribar fins a l'expulsió de la xarxa.

VISA

Una de les xarxes d'identificació més famoses és la xarxa VISA, que s'encarrega de federar la identitat de targetes de crèdit entre entitats bancàries. De fet, VISA és la xarxa de federació de pagaments més gran que hi ha, amb desenes de milers d'afiliats arreu del món.

Aquesta xarxa va sorgir a partir de les primeres emissions de targetes de crèdit que va fer el Bank of America, anomenada *Bank Americard*. Després de passar per diferents etapes, entre les quals hi ha la federació *ad hoc* i la *hub and spoke*, es va crear la xarxa d'identitats que avui coneixem com a *VISA*, que és la xarxa de federació de targetes de crèdit més gran del món, i que ha donat resposta als diferents problemes que hi havia en els dos patrons de federació esmentats abans.

1.4. *Single sign-on*

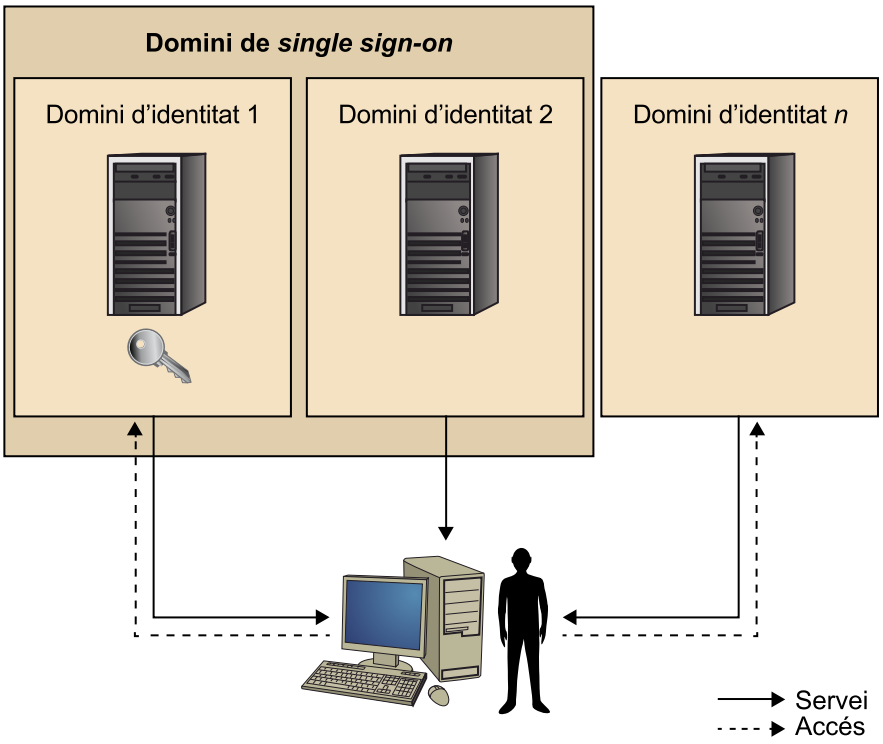
Dins de tots els serveis per a la identificació i la gestió d'accés que hem comentat més amunt, l'autenticació ha estat el que ha gaudit més dels beneficis de la federació i ha arribat, fins i tot, a associar federació amb *single sign-on*.

El *single sign-on*, o autenticació única, es basa en el fet que l'usuari s'autentica una única vegada en un dels sistemes federats i, a partir d'aquí, pot accedir a la resta de sistemes sense haver-se de tornar a autenticar.

Operacions de *single sign-off*

De la mateixa manera que hi ha operacions de *single sign-on*, també n'hi ha de *single sign-off*, imprescindible també per a evitar, per exemple, segrestos de sessió.

Diagrama que mostra el procés d'autenticació únic per a diferents dominis d'identitat



A favor d'aquest mecanisme hi ha la reducció del nombre de credencials que cal recordar (si són del tipus usuari/contrasenya), de manera que disminueix el que s'ha anomenat *fatiga d'autenticació*. També permet accelerar els processos d'accés, ja que no requereix intervenció humana. Permet gestionar els accessos centralitzadament i reduir els costos de manteniment global.

En contra d'aquest mecanisme tenim que, una vegada s'han robat les credencials d'un usuari, immediatament es té accés a molts sistemes sense cap mecanisme de seguretat que ho impedeixi. Per a mitigar aquest fet, com que únicament s'ha de recordar una única contrasenya, en aquest tipus de sistemes les polítiques de generació de contrasenyes solen ser més estrictes.

Els sistemes d'identitat, necessaris per a qualsevol procés de federació, solen ser sistemes d'una mateixa organització que fins i tot moltes vegades deleguen els processos d'identificació a tercers, de manera que s'esvaeix llavors el concepte inicial de *federació* cap a un concepte d'*autenticació delegada*.

Vegeu també

Tractarem del concepte d'*autenticació delegada* en el subapartat 3.7 d'aquest mòdul.

Finalment, cal comentar que s'han generat molts acostaments per al servei de *single sign-on*, encara que són els estàndards més desenvolupats els que s'acaben imposant a l'hora de fer desplegaments d'una certa entitat. Fins i tot en aquest grup d'estàndards, cadascun ha nascut i crescut amb un objectiu al cap, de manera que s'adequa millor a les missions que coincideixen amb l'enfocament pel qual es van concebre.

2. Estàndards

Com hem vist més amunt, la federació d'identitats es basa en l'intercanvi d'informació entre diferents dominis d'identitat. Per a això, és imprescindible tenir estàndards adequats que permetin un llenguatge comú entre sistemes de diferent índole.

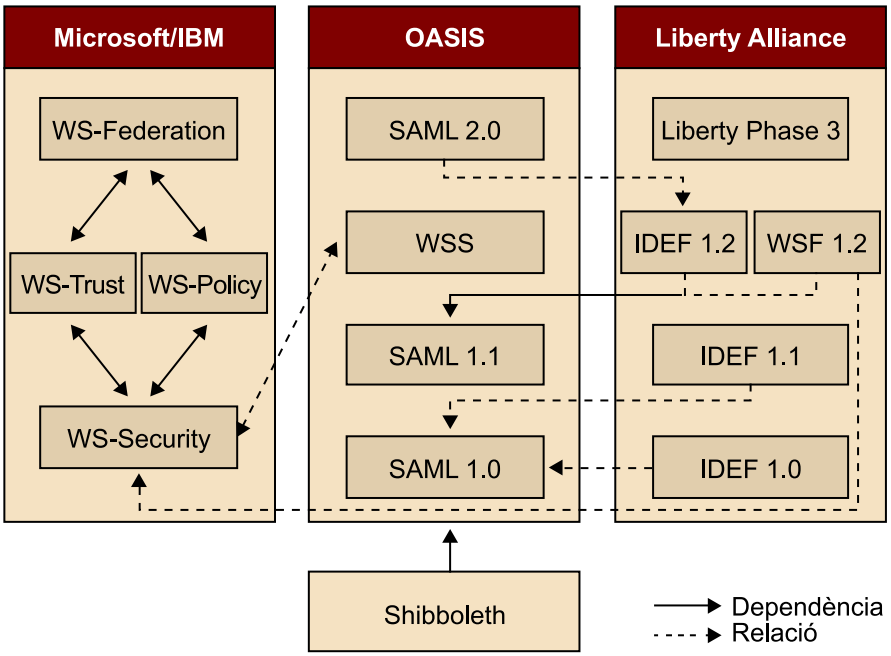
No és d'estranyar que hagin sorgit diferents iniciatives per a donar resposta a aquestes necessitats, generalment a partir d'associacions o unions empresarials de gran impacte en el món de les tecnologies de la informació i la comunicació.

Dins d'aquests estàndards ens centrarem bàsicament en quatre:

- Microsoft/IBM
- OASIS
- Liberty Alliance
- Shibboleth (Internet 2)

La figura mostra la relació que hi ha entre els diferents estàndards i els protocols que hi estan associats.

Diagrama de relació entre estàndards



Sembla difícil saber quin es convertirà en l'estàndard *de facto* en el mercat. La tendència actual apunta al fet que SAML ha estat adoptat, d'una manera o una altra, per Microsoft/IBM i per Liberty Alliance, però no es deixa entreveure quina serà la relació entre Liberty Alliance i Microsoft/IBM.

2.1. Microsoft/IBM

Al principi de la dècada del dos mil, Microsoft i IBM van elaborar un paper[4] conjunt per crear *web services* segurs. Aquesta especificació es va anomenar *WS-** i està basada en la creació d'una sèrie de testimonis o *tokens* que s'adjunten als missatges, algun dels quals té a veure amb la identitat.

(4)De l'anglès, *paper*.

S'estructura de manera modular, aïllant components de la seguretat diferenciats en diferents especificacions. A continuació descrivim els més importants, encara que hi ha diferents especificacions WS-* per a resoldre diferents aspectes:

- **WS-Policy**. És un llenguatge que descriu la política de seguretat d'un determinat servei web o *web service*. Per exemple, com que WS-Security pot utilitzar diferents sistemes de testimonis (com SAML o Kerberos), la política de seguretat podria enunciar l'ús d'SAML 1.1 i no de Kerberos, per exemple.

- **WS-Trust**. És un llenguatge que permet als serveis d'una autoritat en la qual es confia intercanviar un testimoni amb una altra autoritat. Aquest aspecte és la base de la federació en si mateixa.

- **WS-Federation**. És un llenguatge que orquestra les interaccions dels diferents objectes WS-Trust (identitats, atributs, autenticacions, etc.) que participen en un servei web.

2.2. OASIS

SAML va ser creat per l'Organització per a l'Avenç dels Estàndards de la Informació Estructurada (OASIS), que és un consorci internacional sense ànim de lucre que produeix estàndards tecnològics de negoci. A més d'aquest llenguatge, el més àmpliament utilitzat i conegut, n'ha produït altres també de molt populars, com el *service provisioning markup language* (SPML) i l'*extensible access control markup language* (XACML).

En general, els sistemes d'automatització de la identitat necessiten una manera de distribuir les assercions d'autenticació i autorització, com es fa per exemple amb Kerberos. SAML, en aquest aspecte, ha guanyat adeptes en els últims anys i es deixa entreveure com l'estàndard *de facto* en un futur.

Vegeu també

SAML i Kerberos els estudiarem, respectivament, en els subapartats 3.1 i 3.7 d'aquest mòdul.

SAML es basa essencialment en un client que demana una cosa sobre una identitat a una autoritat SAML[5], que al seu torn respon amb el que s'ha anomenat *assercions*. Es poden tenir tres tipus d'assercions, i per tant tres tipus d'autoritats SAML:

(5)En l'apartat 3 veurem amb més detall els mecanismes que hi estan involucrats.

1) Assercions d'autenticació. En aquest cas el client demana sobre l'autenticació del subjecte **S** de la forma **F** en el temps **T**.

Per exemple, una asserció d'aquest tipus és la següent: "Alice a micompania.com es va autenticar amb contrasenya el 20011-07-20T17:00:00-05:00".

2) Assercions d'atribut. Una vegada s'ha autenticat el subjecte **S**, el client pot preguntar una sèrie d'atributs que estan associats a aquest subjecte, amb els valors corresponents.

Per exemple, "Alice està associada amb l'atribut Departament amb valor Enginyeria i amb l'atribut correu electrònic i valor alice@micompania.com".

3) Assercions d'autorització. Quan el subjecte **S** vol accedir a un determinat recurs, el client pregunta a *un policy decission point* (servidor SAML d'assercions d'autorització) per la possibilitat d'accedir-hi duent a terme l'acció **A**.

Per exemple, "El subjecte http://boo.com/foo té permís per a llegir http://b.com/bar evidenciat per les assercions A1, A2 i A8".

Els tres tipus d'assercions no han de ser emesos necessàriament per tres serveis diferents. Un servei pot emetre els tres tipus d'assercions, alhora que un servei d'assercions pot ser client i preguntar a un altre servei SAML.

Generalment, una asserció conté:

- ID[6] de l'emissor i data de temps d'expedició.
- ID únic de l'asserció.
- Subjecte, incloent-hi el nom del domini de seguretat i, opcionalment, les dades d'autenticació.
- Perfil d'informació addicional de l'entitat emissora anomenada *advine*.
- Condicions per les quals l'asserció és vàlida (per exemple, data de caducitat).
- Restriccions d'audiència.
- Restriccions de destinació, com el recurs específic (per exemple, URL).
- Condicions específiques de l'aplicació.

(6)La sigla *ID* correspon al terme *identificador*.

Vegeu també

En el subapartat 3.1 es mostren amb més detall les particularitats d'SAML, sens dubte el llenguatge de federació més adoptat en aquests moments per la indústria, encara que la majoria de productes ofereixen una gran compatibilitat amb molts altres estàndards de federació.

2.3. Liberty Alliance

Liberty Alliance és un consorci de més de cent setanta empreses per a la creació d'estàndards i especificacions per a la federació d'identitats. Al començament, el consorci volia crear una única especificació per a la federació d'identitats, però finalment la van transformar en tres especificacions diferents:

Enllaç d'interès

Per a saber més coses sobre el projecte Liberty Alliance, consulteu el web http://www.projectliberty.org.

1) Identity Federation Framework (ID-FF). Permet el *single sign-on* i el vincle de comptes entre participants que tenen una relació de confiança.

2) Identity Web Services Framework (ID-WSF). Permet que grups de participants de confiança entre si es puguin unir a altres grups, i permet als usuaris el control sobre la manera com es comparteix aquesta informació.

3) Identity Services Interface Specifications (OD-SIS). S'utilitza per a construir un conjunt de serveis interoperables sobre ID-WSF.

2.4. Internet 2 i Shibboleth

Internet 2 és una coalició d'universitats americanes que treballen per crear la pròxima generació d'Internet, i Shibboleth és la seva estructura arquitectònica per a la gestió d'identitats. Shibboleth està basat per complet en SAML, i proveeix bàsicament de serveis de *single sign-on* i administració federada de l'accés a recursos restringits.

Com que la seva negociació està basada en atributs, és un protocol que té per disseny mecanismes per a garantir la privadesa. Això últim contribueix a una tendència recent de tenir en compte el disseny dels sistemes a la seguretat (*security by design*) i a la privadesa (*privacy by design*).

3. Tecnologia per a la gestió d'identitats federades

L'objectiu d'aquest apartat és repassar els estàndards que s'adopten més en la indústria per a cobrir les necessitats de la gestió d'identitats federades: des del control d'accés fins a l'autenticació, passant per la distribució de polítiques o la provisió d'identitats.

3.1. SAML

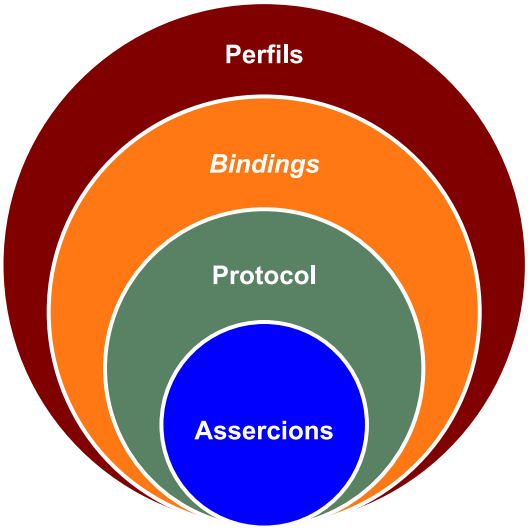
Security Assertion Markup Language (SAML) és un estàndard d'OASIS basat en XML per a l'intercanvi d'informació d'autenticació i autorització entre dominis de seguretat.

L'especificació SAML 2 està dividida en els components que es mostren en la figura següent:

Versions d'SAML

Actualment, hi ha dues versions diferents actives, SAML 1.1 i SAML 2, encara que aquí ens centrarem en l'última versió, la 2.0.

Diagrama de components d'SAML 2.0



3.1.1. Assercions SAML

Una asserció SAML és una unitat d'informació emesa en un moment determinat per una autoritat SAML o emissor, sobre un subjecte acollit a determinades condicions, i que queda definida per l'estructura següent:

```
<saml:Assertion ...>
...
</saml:Assertion>
```

Dins d'una asserció, hi podem afegir tres tipus de sentències:

1) Autenticació. Usem aquesta sentència quan volem afirmar que un usuari es va autenticar per mitjà d'un mecanisme i en un moment determinat. Només s'obté informació sobre l'acció d'autenticació, no sobre la realització d'aquesta acció.

2) Atribut. Usem aquesta sentència quan volem establir una relació entre un determinat usuari i una sèrie d'atributs.

3) Autorització. Usem aquesta sentència per a informar si acceptem o denegem l'accés d'un usuari a un recurs protegit. A més, podem rebre informació extra en una petició de decisió d'autorització que l'emissor hagi considerat determinant a l'hora d'avaluar la nostra resposta.

Així mateix, una asserció pot contenir més d'un tipus de sentència, de manera que és comú trobar-se assercions amb sentències d'autenticació i d'atributs, com veiem en l'exemple següent:

Exemple

```
<saml2:AssertionID="hgd1jcigrccv9osuk564do272i" IssueInstant=
 "2009-10- 08T16:44:50Z" Version="2.0">
<saml2:Issuer>http://localhost/exampleniflocator</saml2:Issuer>
<saml2:Subject>
<saml2:NameID
   Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
   andres
</saml2:NameID>
<saml2:SubjectConfirmation
   Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>
</saml2:Subject>
<saml2:Conditions NotBefore="2009-10-08T16:39:50Z"
   NotOnOrAfter="2009-10-08T16:54:50Z"/>
<saml2:AuthnStatement AuthnInstant="2009-10-08T16:44:50Z">
<saml2:AuthnContext>
<saml2:AuthnContextClassRef>
   urn:oasis:names:tc:SAML:2.0:ac:classes:Password
</saml2:AuthnContextClassRef>
</saml2:AuthnContext>
</saml2:AuthnStatement>
<saml2:AttributeStatement>
<saml2:Attribute Name="NifNie">
<saml2:AttributeValue>28806345S</saml2:AttributeValue>
</saml2:Attribute>
</saml2:AttributeStatement>
</saml2:Assertion>
```

L'estructura de l'asserció SAML mostrada és la següent:

- Emissor de l'asserció, mitjançant l'element `<saml2:Issuer>`, que indica qui l'ha generada mitjançant una cadena de caràcters. Normalment és un URI[7].
- Subjecte, mitjançant l'element `<saml2:Subject>`, que indica sobre quin *principal* o usuari s'ha generat l'asserció i el format en què s'especifica aquesta informació.
- Condicions, amb l'element `<saml2:Conditions>`, que indica des de quin moment és vàlida l'asserció i fins quan.
- Sentències d'atributs i autenticació.

(7)La sigla *URI* correspon a *identificador uniforme de recursos* o *uniform resource identifier*.

Exemple

En la classe Java següent, veurem com es crea una asserció SAML amb una estructura similar a la que hem vist en l'exemple anterior:

```
import com.sun.xml.wss.saml.*;
import java.io.StringWriter;
import java.util.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import org.w3c.dom.Document;

public class Example2 {

public static void main(String[] args) throws Exception {
SAMLAssertionFactory af = SAMLAssertionFactory.newInstance(
SAMLAssertionFactory.SAML2_0);

String assertionId = UUID.randomUUID().toString();

NameID nameId = af.createNameID("cr01", null, null);

GregorianCalendar issuerInst = new GregorianCalendar();

Subject subject = af.createSubject(nameId,
af.createSubjectConfirmation(null,
"urn:oasis:names:tc:SAML:2.0:cm:bearer"));

List<Object> statements = new<Object>LinkedList ();
List attributes = new LinkedList();
String nameAttr = "sn1";
List<String> valuesAttr = new<String>LinkedList ();
valuesAttr.add("Rodriguez");
attributes.add( af.createAttribute(nameAttr, valuesAttr) );
AttributeStatement attrSt = af.createAttributeStatement(attributes);
statements.add(attrSt);
AuthnContext ac = af.createAuthnContext(
"urn:oasis:names:tc:SAML:2.0:ac:classes:Password", null);
AuthnStatement as = af.createAuthnStatement(null, null, ac, null,
null);
statements.add(as);
Assertion assertion = af.createAssertion
(assertionId, nameId, issuerInst, null, null, subject, statements);
showAssertion(assertion);
}
}
```

En aquest exemple, usem les classes que ens ofereix la biblioteca Metrov1.4 per a treballar amb assercions SAML. Com veiem, tenim pràcticament una classe per element que apareix dins d'una asserció.

La classe `SAMLAssertionFactory` ens permet crear, de manera segura, cadascun dels elements que necessitem. L'aspecte principal que s'ha de tenir en compte usant aquesta factoria és que permet generar elements per SAML 1 i SAML 2; com que les assercions i els elements que hi ha a dins són diferents entre aquestes versions, només s'han d'usar els mètodes adequats.

Per a crear una asserció SAML 2, hem de definir els elements següents:

- Identificador per a l'asserció; en aquest cas, el generem dinàmicament:

```
String assertionId = UUID.randomUUID().toString();
```

- Identificador de nom, que identifica el subjecte sobre el qual s'emet l'asserció:

```
NameID nameId = af.createNameID("cr01", null, null);
```

- Subjecte, que conté tant l'identificador de nom com la informació de la manera en què el receptor de l'asserció ha de provar la identitat de l'usuari:

```
Subject subject = af.createSubject(nameId,
af.createSubjectConfirmation(null,
```

```
        "urn:oasis:names:tc:SAML:2.0:cm:bearer"));
```

- Instant en què s'emet l'asserció:

```
GregorianCalendar issuerInst = new GregorianCalendar();
```

- Sentència d'autenticació, que indica com s'ha comprovat la identitat de l'usuari:

```
AuthnContext ac = af.createAuthnContext(
      "urn:oasis:names:tc:SAML:2.0:ac:classes:Password", null);
      AuthnStatement as = af.createAuthnStatement(null, null, ac,
null, null);
```

- Sentència d'atributs, amb la llista d'atributs que estan assignats per a aquest subjecte:

```
List attributes = new LinkedList();
String nameAttr = "sn1";
List<String> valuesAttr = new<String>LinkedList ();
valuesAttr.add("Rodriguez");
attributes.add( af.createAttribute(nameAttr, valuesAttr) );
AttributeStatement attrSt = af.createAttributeStatement(attributes);
```

3.1.2. Protocol SAML

SAML defineix un protocol d'intercanvi de missatges del tipus petició-resposta (*request-response*) amb l'objectiu d'obtenir assercions d'un usuari.

D'aquesta manera, una petició SAML pot sol·licitar informació sobre els elements següents:

- Autenticació
- Decisió d'autorització
- Atributs

Cada tipus de petició està orientat a obtenir sentències que corresponen a cadascun d'aquests tipus. Aquestes sentències són usades perquè, en l'especificació SAML 2, es defineixin els protocols següents:

- **Petició i consulta d'atributs**. Amb aquest protocol es pot sol·licitar una asserció utilitzant-ne la referència i també es poden consultar els atributs d'un subjecte.

- **Petició d'autenticació**. Gràcies a aquest protocol es pot iniciar el procés d'autenticació del subjecte amb l'objectiu d'obtenir assercions que l'identifiquin.

- **Resolució d'artefacte**. L'especificació SAML permet enviar una referència, anomenada *artefacte*, en lloc d'un missatge. D'aquesta manera, quan un receptor obté aquest artefacte, pot utilitzar el servei de resolució d'artefactes de l'entitat que l'hagi emès per a obtenir el missatge al qual representava.

- **Gestió d'identificadors de nom** (o *name identifier*). Permet gestionar, en determinats aspectes, l'identificador de nom d'un subjecte en un proveïdor d'identitat.

- **Final de sessió simple** (o *logout* simple). Permet acabar la sessió d'un subjecte en un domini d'identitat d'un proveïdor de servei.

- **Assignacions d'identificadors de nom**. Permeten sol·licitar la conversió del format que s'utilitza a l'hora d'expressar un identificador de nom.

De tots plegats, el més usat és el protocol de petició i consulta d'atributs.

Exemple

L'exemple següent mostra l'estructura típica d'una petició d'atribut:

```
<samlp:AttributeQueryxmlns:samlp='urn:oasis:names:tc:SAML:2.0:
   protocol'
   ID='6985f6b9-8946-4c9f-9dac-4180ea3f6c88'
   IssueInstant='2009-11-22T11:20:35Z' Version='2.0'>
<saml:Issuer xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>
   http://www.prise.es/uoc-book/example2
</saml:Issuer>
<saml:Subject xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>
<saml:NameID
   Format='urn:oasis:names:tc:SAML:2.0:nameid-format:unspecified'
   xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>
   cr01
</saml:NameID>
</saml:Subject>
</samlp:AttributeQuery>
```

Per a generar una petició d'atribut com la que hem mostrat en aquest exemple, hem d'utilitzar la classe Java següent:

```
import java.io.StringReader;
import java.text.SimpleDateFormat;
import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dg.*;
import org.xml.sax.InputSource;
public class Example2 {
private static final String reqMsg =
"<samlp:AttributeQuery " +
" xmlns:samlp='urn:oasis:names:tc:SAML:2.0:protocol' " +
" ID='%%ID%%' IssueInstant='%%ISSUEINST%%' Version='2.0'>" +
" <saml:Issuer xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>"
+
"%%ISSUER%%" +
" </saml:Issuer>" +
" <saml:Subject xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>"
 +
" <saml:NameID Format='urn:oasis:names:tc:SAML:2.0:nameid-format:
 unspecified'" +
" xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'> " +
"%%SUBJECT%%" +
" </saml:NameID>" +
" </saml:Subject>" +
"</samlp:AttributeQuery>";
public static String getAttributeRequest(String id, String issuer,
String subject) {
Calendar cal = Calendar.getInstance(TimeZone.getTimeZone("US/
Arizona"));
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
SimpleDateFormat sdfTime = new SimpleDateFormat("HH:mm:ss");
String res = Example2.reqMsg
```

```
res = res.replaceAll("%%ID%%", id);
res = res.replaceAll("%%ISSUEINST%%", sdf.format(cal.getTime()) +
"T" +
sdfTime.format(cal.getTime()) + "Z");
res = res.replaceAll("%%ISSUER%%", issuer);
res = res.replaceAll("%%SUBJECT%%", subject);
return res;
}

public static Element getElementFromString(String s) throws
Exception {
DocumentBuilderFactory factory = DocumentBuilderFactory.
newInstance();
factory.setNamespaceAware(true);
DocumentBuilder builder = factory.newDocumentBuilder();
InputSource is = new InputSource(new StringReader(s));
Document d = builder.parse(is);
return d.getDocumentElement();
}
public static void main(String[] args) throws Exception {
String id = UUID.randomUUID().toString();
String issuer = "http://www.prise.es/uoc-book/example2";
String subject = "cr01";
String reqMsg = getAttributeRequest(id, issuer, subject);
System.out.println(reqMsg);
Element req = getElementFromString(reqMsg);
}
}
```

Com veiem en el mètode `getAttributeRequest`, una de les maneres més còmodes de treballar amb SAML és representar el missatge directament en un objecte tipus `String`, ja que redueix complexitats a l'hora d'importar biblioteques al nostre projecte. Malgrat això, evidentment s'ha d'anar amb compte amb l'estructura del missatge, ja que ens podem confondre i pot ser que no utilitzem correctament els elements permesos i els valors d'aquests elements en el format adequat en el missatge que volem transmetre.

Una vegada tenim el missatge complet, hem d'utilitzar la funció `getElementFromString` per a obtenir un objecte del tipus `org.w3c.dom.Element` que ens resultarà més fàcil d'integrar en peticions SOAP[8] en un futur.

[8] *SOAP* és la sigla de *protocol d'accés a objecte simple* o *simple object access protocol.*

En les peticions d'atributs podem consultar tots els atributs de l'usuari, per a la qual cosa no indiquem cap llista d'atributs en la petició, o bé podem sol·licitar uns atributs determinats, indicant en la petició quins volem consultar.

Una petició d'atributs és resposta per una **resposta SAML** `<samlp:Response>`, que té l'estructura següent:

- Emissor de la resposta SAML, per mitjà de l'element `<saml:Issuer>`.
- Tipus de resposta, que informa si hi ha hagut algun d'error a l'hora de processar la petició o generar la resposta, reflectit en l'element `<samlp:Status>`.
- Una asserció SAML o més d'una, l'aparició de la qual és opcional.

Per al cas de la petició d'atributs, s'hi ha d'incloure una asserció com la que hem mostrat en el primer exemple, en la qual, dins d'una sentència d'atributs, s'han d'incloure els atributs que sí que té l'usuari i que l'entitat SAML, generalment el servei d'autoritat d'atributs (AA) del proveïdor d'identitat, ha autoritzat d'emetre a aquest emissor de la petició, depenent de la seva política d'emissió d'atributs.

Autoritat i assercions d'atributs

Una autoritat d'atributs és una entitat del sistema que produeix les assercions d'atributs. Una asserció d'atributs és una asserció que transmet informació sobre els atributs d'un subjecte. Un atribut és una nota característica d'un objecte.

Exemple

```
<samlp:Response xmlns:samlp="..." xmlns:saml="..." xmlns:ds="..."
ID="_6c3a4f8b9c2d" Version="2.0" IssueInstant="2004-03-27T08:42:
00Z">
<saml:Issuer>https://www.example.com/saml</saml:Issuer>
<ds:Signature> ... </ds:Signature>
<Status>
<StatusCode Value="..."/>
</Status>
<saml:Assertion>
...
<saml:Assertion>
</samlp:Response>
```

L'atribut `Value` de l'element `<StatusCode>` informa del tipus de resposta que es retorna i, encara que l'especificació de SAML 2 defineix una nombrosa llista de valors amb la semàntica d'aquests valors, destaquem els següents:

- `urn:oasis:names:tc:SAML:2.0:status:Success`, en cas que la petició hagi estat processada correctament.
- `urn:oasis:names:tc:SAML:2.0:status:AuthnFaile`, si no s'ha pogut verificar la identitat del subjecte.
- `urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue`, si hi ha hagut un error en processar el nom o el valor d'algun dels atributs.

3.1.3. SAML *bindings*

En SAML, s'anomena *binding* el procediment d'assignar un protocol SAML, com el de petició-resposta que hem vist més amunt, a una capa de transport que permeti fer-ne la transmissió.

Els *bindings* disponibles en SAML 2 són els següents:

- **SOAP 1.1**. Permet enviar i rebre missatges SAML utilitzant SOAP 1.1.
- **SOAP invertit** o **PAOS**. Aquest *binding* permet respondre a un client HTTP que intenta accedir a un recurs amb un missatge SOAP que, al seu torn, inclou una petició SAML. D'aquesta manera, aquest client HTTP processa la petició i respon amb una resposta SAML mitjançant SOAP. Si aquesta resposta és correcta, el receptor permet l'accés d'aquest client HTTP al recurs protegit.

- **Redireccions HTTP**. Aquest *binding* permet enviar missatges del protocol SAML mitjançant els paràmetres d'un URL.

- **Missatges HTTP amb mètode POST**. Amb aquest mètode es poden transmetre missatges SAML codificats en base 64 mitjançant un formulari HTML.

- **Artefactes en missatges HTTP**. En aquest *binding* es transmet tant la petició SAML com la resposta per mitjà d'un artefacte. Així, quan rebem un artefacte, hem d'utilitzar un altre dels *bindings* per a resoldre aquest artefacte.

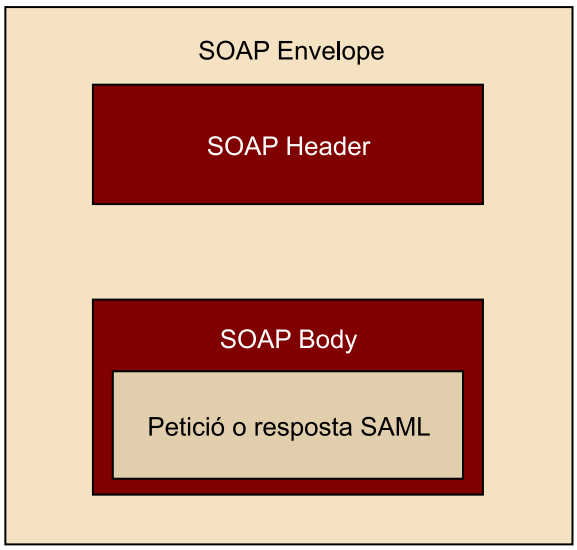
- **URI**. Referencia una asserció SAML amb un identificador independent del mitjà de transport, que pot ser inclòs en elements com `<saml:AssertionIDRef>`.

Nosaltres ens centrarem en els *bindings* SOAP 1.1 i missatges HTTP amb mètodes POST, ja que són els més comuns dins de les infraestructures d'autenticació i autorització.

3.1.4. *Binding* SOAP 1.1

Com hem dit més amunt, el *binding* SOAP 1.1 permet enviar i rebre peticions-respostes SAML per mitjà del protocol SOAP 1.1.

Diagrama de components d'un missatge SAML



Exemple

En l'exemple següent veiem una petició HTTP d'una sol·licitud d'atribut mitjançant SOAP 1.1.

```
POST /SamlService HTTP/1.1
Host: www.prise.es
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security
<SOAP-ENV:Envelope
```

```
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<samlp:AttributeQuery
   xmlns:samlp='urn:oasis:names:tc:SAML:2.0:protocol'
   ID='6985f6b9-8946-4c9f-9dac-4180ea3f6c88'
   IssueInstant='2009-11-22T11:20:35Z' Version='2.0'>
   ...
   </samlp:AttributeQuery>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Com veiem, tan sols hem d'encapsular la nostra petició d'atributs anterior dins del cos del missatge SOAP. En l'exemple següent s'observa la creació de la petició d'atributs amb SOAP 1.1 en Java.

```
import java.io.StringReader;
import java.text.SimpleDateFormat;
import java.util.*;
import javax.xml.parsers.*;
import javax.xml.soap.*;
import org.w3c.dg.*;
import org.xml.sax.InputSource;

public class Example3 {
private static final String reqMsg =
   "<samlp:AttributeQuery " +
   " xmlns:samlp='urn:oasis:names:tc:SAML:2.0:protocol' " +
   " ID='%%ID%%' IssueInstant='%%ISSUEINST%%' Version='2.0'>" +
   " <saml:Issuer xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>"
    +
   "%%ISSUER%%" +
   " </saml:Issuer>" +
   " <saml:Subject xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'>"
    +
   " <saml:NameID Format='urn:oasis:names:tc:SAML:2.0:nameid-format:
    unspecified'" +
   " xmlns:saml='urn:oasis:names:tc:SAML:2.0:assertion'> " +
   "%%SUBJECT%%" +
   " </saml:NameID>" +
   " </saml:Subject>" +
   "</samlp:AttributeQuery>";

public static String getAttributeRequest(String id, String issuer,
   String subject) {
   Calendar cal =
   Calendar.getInstance(TimeZone.getTimeZone("US/Arizona"));
   SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
   SimpleDateFormat sdfTime = new SimpleDateFormat("HH:mm:ss");
   String res = Example3.reqMsg;

   res = res.replaceAll("%%ID%%", id);
   res = res.replaceAll("%%ISSUEINST%%", sdf.format(cal.getTime()) +
   "T" +
   sdfTime.format(cal.getTime()) + "Z");
   res = res.replaceAll("%%ISSUER%%", issuer);
   res = res.replaceAll("%%SUBJECT%%", subject);
   return res;
}

public static Element getElementFromString(String s) throws
   Exception {
   DocumentBuilderFactory factory =
   DocumentBuilderFactory.newInstance();
   factory.setNamespaceAware(true);
   DocumentBuilder builder = factory.newDocumentBuilder();

   InputSource is = new InputSource(new StringReader(s));
   Document d = builder.parse(is);

   return d.getDocumentElement();
}
```

```
public static void main(String[] args) throws Exception {
   String id = UUID.randomUUID().toString();
   String issuer = "http://www.prise.es/uoc-book/example2";
   String subject = "cr01";
   String reqMsg = getAttributeRequest(id, issuer, subject);
   Element req = getElementFromString(reqMsg);

   MessageFactory messageFactory = MessageFactory.newInstance();
   SOAPMessage message = messageFactory.createMessage();
   SOAPPart soapPart = message.getSOAPPart();
   SOAPEnvelope envelope = soapPart.getEnvelope();
   SOAPBody body = envelope.getBody();
   body.addDocument(req.getOwnerDocument());
   message.saveChanges();

   SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
   SOAPConnection sc = scf.createConnection();
   SOAPMessage reply = sc.call(message,
   "http://www.exemple.com/samlservice");
}
}
```

Com veiem, dins del mètode estàtic `main`, una vegada tenim la petició d'atributs en un objecte `org.w3c.dom.Element`, generem el missatge SOAP i afegim dins del cos aquesta petició:

```
MessageFactory messageFactory = MessageFactory.newInstance();
   SOAPMessage message = messageFactory.createMessage();
   SOAPPart soapPart = message.getSOAPPart();
   SOAPEnvelope envelope = soapPart.getEnvelope();
   SOAPBody body = envelope.getBody();
   body.addDocument(req.getOwnerDocument());
message.saveChanges();
```

D'aquesta manera, ja tenim una petició SOAP amb una capçalera buida –ja que no cal afegir cap element per a aquest exemple– i un cos amb la petició d'atributs. Tan sols ens falta enviar aquesta petició:

```
SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
SOAPConnection sc = scf.createConnection();
SOAPMessage reply = sc.call(message,
   "http://www.exemple.com/samlservice");
```

El missatge queda enviat al servei desplegat a http://www.exemple.com/samlservice i obtenim la resposta en la variable `reply`.

En cas que vulguem mostrar, per consola, quina ha estat la resposta, hi hem d'afegir les ordres següents:

```
TransformerFactory transformerFactory = TransformerFactory.
 newInstance();
Transformer transformer =
   transformerFactory.newTransformer();
Source sourceContent = reply.getSOAPPart().getContent();
StreamResult result = new StreamResult(System.out);
transformer.transform(sourceContent, result);
```

Una vegada enviada la petició a aquest servei, aquest servei mateix processa la informació i retorna una resposta SAML dins d'un missatge SOAP:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<samlp:Response ...>
   ...
</samlp:Response>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

Com observem, l'element `<samlp:Response>`, que representa la resposta de la petició SAML rebuda, s'inclou dins del cos d'un missatge SOAP.

3.1.5. *Binding* missatge HTTP amb mètode POST

El *binding* missatge HTTP amb mètode POST s'utilitza en els casos en què l'emissor i el receptor del missatge SAML han d'utilitzar el navegador de l'usuari com a intermediari.

Els missatges SAML es codifiquen dins d'un formulari HTML i s'envien mitjançant mètode POST.

En cas que el missatge SAML sigui una petició, l'element que conté el missatge SAML dins del formulari es diu `SAMLRequest`, mentre que si és una resposta es diu `SAMLResponse`.

També se sol incloure en el formulari HTML, com un camp ocult, el paràmetre opcional `RelayState`. Aquest paràmetre s'utilitza per a facilitar la gestió de l'estat de la informació i en aquest *binding* no s'han de superar els vuitanta caràcters. En cas que s'enviï un `RelayState`, l'element que genera el missatge SAML de resposta ha d'incloure el mateix valor per a la resposta, mentre que si no se n'envia cap en la petició, en generar la resposta es pot establir un valor al `RelayState` sempre que hi hagi un acord previ en l'ús del *binding*.

Exemple

A continuació, presentem un exemple d'enviament de petició SAML amb *binding* HTTP Post:

```
<form method='post' action='SamlService'>
<input type='hidden' name='RelayState'
   value='http://www.prise.es/uoc-book/SamlService' />
<input type='hidden' name='SAMLRequest' value='PEVudGVyIHRleH...' />
</form>
```

Així, en aquest exemple, incloem una petició SAML, que queda enviada mitjançant aquest *binding*, ja que tant aquesta petició com el `RelayState` queden associats al formulari que s'ha generat i que ha enviat el navegador. A més, en la majoria dels casos, el formulari s'envia automàticament gràcies al fet que, amb JavaScript, s'indica que s'enviï en carregar aquesta pàgina.

Per a processar la petició, podem desenvolupar una senzilla miniaplicació de servidor o *servlet*, que comprovi si han arribat els dos paràmetres i enviï l'asserció com a resposta a l'URL enviat en el camp `RelayState`:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Map;
import javax.servlet.ServletException;
import javax.servlet.http.*;
import sun.misc.BASE64Decoder;

public class Example4 extends HttpServlet {

private String processRequest(String samlRequest) { ... }
```

```
protected void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
PrintWriter out = response.getWriter();

Map params = request.getParameterMap();
if (params.containsKey("SAMLRequest") &&
params.containsKey("RelayState")) {
String relayState = request.getParameter("RelayState");
String samlReq = request.getParameter("SAMLRequest");

BASE64Decoder decoder = new BASE64Decoder();
byte[] decodedBytes = decoder.decodeBuffer(samlReq);

String samlRequest = new String(decodedBytes);
String samlResponse = processRequest(samlRequest);
out.println("<html>");
out.println("<head><title>Servlet</title></head>");
out.println("<body>");
out.println("<form method='post' action='"+relayState+"'>");
out.println("<input type='hidden' name='SAMLResponse' value='"+
 samlResponse+"' />");
out.println("<input type='hidden' name='RelayState' value='"+
 relayState+"' />");
out.println("<input type='submit' name='submit' value='Acceptar'/>");
out.println("</form>");
out.println("</body>");
out.println("</html>");
}
}
}
```

Deixem de banda el codi del mètode `processRequest`, ja que la seva lògica s'ha explicat en els exemples anteriors, i ens centrem en la manera com hem de comprovar els paràmetres i generar la resposta al navegador o agent de l'usuari:

```
Map params = request.getParameterMap();
if (params.containsKey("SAMLRequest") &&
params.containsKey("RelayState")) {
   String relayState = request.getParameter("RelayState");
   String samlReq = request.getParameter("SAMLRequest");
   BASE64Decoder decoder = new BASE64Decoder();
   byte[] decodedBytes = decoder.decodeBuffer(samlReq);
```

El primer pas és comprovar que hi ha tots els paràmetres que s'esperaven en el missatge rebut per mètode POST, i descodificar la petició SAML en base 64:

```
String samlRequest = new String(decodedBytes);
String samlResponse = processRequest(samlRequest);
```

La lògica principal d'aquesta miniaplicació de servidor és processar aquesta petició SAML i generar una resposta:

```
out.println("<html>");
out.println("<head><title>Servlet</title></head>");
out.println("<body>");
out.println("<form method='post' action='"+relayState+"'>");
out.println("<input type='hidden' name='SAMLResponse' value='"+
 samlResponse+"' />");
out.println("<input type='hidden' name='RelayState' value='"+
 relayState+"' />");
out.println("<input type='submit' name='submit' value='Acceptar'/>");
out.println("</form>");
out.println("</body>");
out.println("</html>");
```

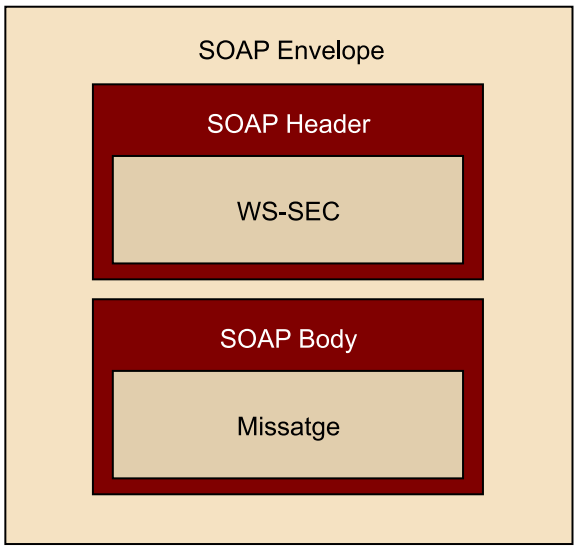
Finalment, generem un altre formulari que s'encarrega d'enviar la resposta SAML a l'URL indicat pel paràmetre `RelayState` mitjançant POST.

3.2. Seguretat en serveis web

L'estàndard *web services security* (W-Sec o WSS), publicat per OASIS, permet dotar els missatges SOAP transmesos cap a serveis web d'una capa d'autenticació i autorització.

Per mitjà de l'element `<wsse:Security>`, que queda associat a la capçalera SOAP del missatge, hi incloem els testimonis de seguretat que contenen informació que el receptor final, o un d'intermediari, del missatge utilitza per a autenticar o verificar el missatge rebut.

Diagrama d'un missatge W-Sec



En una capçalera SOAP, hi pot haver l'element `<wsse:Security>` més d'una vegada, encara que ha d'especificar a qui va dirigit cadascun d'aquests elements mitjançant l'atribut actor. Un altre atribut comú en aquest element és `mustUnderstand`, que permet indicar si el receptor del missatge ha de processar aquest element de seguretat o no, mitjançant els valors `true` i `false`, respectivament.

```
<S11:Envelope>
<S11:Header>
...
<wsse:Security S11:actor="..." S11:mustUnderstand="...">
...
</wsse:Security>
...
</S11:Header>
...
</S11:Envelope>
```

3.2.1. Testimonis de seguretat

L'especificació W-Sec defineix una àmplia llista de tipus de testimonis de seguretat, que es poden classificar en tres grans grups:

1) Testimoni de nom d'usuari o *username token*, que ens permet indicar el nom d'usuari del sol·licitant i, opcionalment, una contrasenya o una clau compartida per a identificar-lo.

2) Testimoni de seguretat en binari o *binary security token*, que facilita la transmissió de testimonis basats en tiquets Kerberos o certificats digitals X.509.

3) Testimoni XML, per a aquells que estan basats en XML.

Ens centrarem tant en el testimoni de nom d'usuari com en el d'XML.

3.2.2. Testimoni de nom d'usuari

El testimoni de nom d'usuari permet identificar un usuari o emissor del missatge SOAP per mitjà d'un nom d'usuari i una clau compartida o contrasenya. Encara que aquesta última és opcional, es recomana que s'hi inclogui en cas que es vulgui donar fiabilitat a l'autenticació de l'usuari o emissor.

L'estructura general del testimoni de nom d'usuari és la següent:

```
<wsse:UsernameToken wsu:id="token-user">
<wsse:Username> ... </wsse:Username>
<wsse:Password Type="..."> ... </wsse:Password>
<wsse:Nonce EncodingType="..."> ... </wsse:Nonce>
<wsu:Created> ... </wsu:Created>
</wsse:UsernameToken>
```

Com veiem, l'element `<wsse:UsernameToken>` representa el testimoni que conté la informació següent:

- Nom d'usuari, mitjançant l'element `<wsse:Username>`.

- Contrasenya de l'usuari, per mitjà de l'element `<wsse:Password>`, que, a més, ha d'especificar en l'atribut `Type` quin tipus de contrasenya s'hi inclou:
  - En cas que sigui una contrasenya sense cap tipus d'encriptació, s'utilitza el valor `#PasswordText`.
  - D'altra banda, si s'envia l'empremta o *digest* de la contrasenya, s'indica per mitjà del valor `#PasswordDigest`.

- Valor criptogràfic aleatori, en l'element `<wsse:Nonce>`, que ens permet detectar atacs de repetició. Per a incloure-hi aquest valor, se sol utilitzar la codificació base 64. En cas que vulguem utilitzar una altra codificació, hem d'indicar quina s'ha utilitzat en l'atribut `EncodingType` d'aquest valor.

- Quan va ser creat el testimoni mitjançant l'element `<wsu:Created>`.

De tots aquests elements, a l'hora d'enviar un testimoni de nom d'usuari només és obligatori el nom d'usuari.

Exemple

En l'exemple següent, desenvolupat en Java, veiem com s'ha d'incloure aquest tipus de testimoni de seguretat en una capçalera SOAP:

```
import java.io.StringReader;
import java.math.BigInteger;
import java.security.*;
import java.text.SimpleDateFormat;
import java.util.*;
import javax.xml.parsers.*;
import javax.xml.soap.*;
import org.w3c.dg.*;
import org.xml.sax.InputSource;
import sun.misc.BASE64Encoder;

public class Example5 {

public static final String USERNAME_TOKEN = "<wsse:Security xmlns:
 wsse=
'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd' xmlns:wsu=
'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd'>" +
" <wsse:UsernameToken wsu:Id='%%USERTOKENID%%'>" +
" <wsse:Username>%%USERNAME%%</wsse:Username>" +
" <wsse:Password>%%PASSWORD%%</wsse:Password>" +
" <wsse:Nonce>%%NONCE%%</wsse:Nonce>" +
" <wsu:Created>%%CREATED%%</wsu:Created>" +
" </wsse:UsernameToken>" +
"</wsse:Security>";

public static String getNonce() {
SecureRandom rand = null;
try {
rand = SecureRandom.getInstance("SHA1PRNG");
} catch (NoSuchAlgorithmException e) {
throw new IllegalArgumentException(e.toString());
}
rand.setSeed(System.currentTimeMillis());
byte[] nonce = new byte[16];
rand.nextBytes(nonce);
BASE64Encoder b64 = new BASE64Encoder();
return b64.encode(nonce);
}
public static String getUsernameToken(String username, String
password) {
Calendar cal = Calendar.getInstance(TimeZone.getDefault());
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
SimpleDateFormat sdfTime = new SimpleDateFormat("HH:mm:ss");

String res = USERNAME_TOKEN.replace("%%USERNAME%%", username);
res = res.replaceAll("%%PASSWORD%%", password);
res = res.replaceAll("%%NONCE%%", getNonce());
res = res.replaceAll("%%USERTOKENID%%",
new BigInteger(130, new SecureRandom()).toString(32));
res = res.replaceAll("%%CREATED%%", sdf.format(cal.getTime()) + "T" +
```

```
sdfTime.format(cal.getTime()) + "Z");

return res;
}

public static Element getElementFromString(String s) throws
Exception {
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
DocumentBuilder builder = factory.newDocumentBuilder();
InputSource is = new InputSource(new StringReader(s));
Document d = builder.parse(is);
return d.getDocumentElement();
}
public static void main(String[] args) throws Exception {
String reqMsg = "<requestMsg/>";
Element req = getElementFromString(reqMsg);
Element headerElem = getElementFromString(
getUsernameToken("alicia", "m1p4ssw0rd.!"));
MessageFactory messageFactory = MessageFactory.newInstance();
SOAPMessage message = messageFactory.createMessage();
SOAPPart soapPart = message.getSOAPPart();
SOAPEnvelope envelope = soapPart.getEnvelope();
SOAPHeader header = envelope.getHeader();
org.w3c.dom.Node headerNode = header.getOwnerDocument().importNode
(headerElem, true);
header.appendChild(headerNode);
SOAPBody body = envelope.getBody();
body.addDocument(req.getOwnerDocument());
message.saveChanges();
SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
SOAPConnection sc = scf.createConnection();© FUOC · PID_00158713 50
Infraestructures d'autenticació i autorització delegada
SOAPMessage reply = sc.call(message, "http://www.prise.es/service");
}
}
```

Aquest exemple mira d'enviar el missatge `<requestMsg/>` amb un testimoni d'usuari inclòs en la capçalera SOAP.

El primer pas és crear el testimoni d'usuari:

```
Element headerElem = getElementFromString(
   getUsernameToken("alicia", "m1p4ssw0rd.!"));
```

Com veiem en el codi, fa una crida al mètode `getUsernameToken` en la qual indica que el nom d'usuari és `alicia` i la contrasenya és `m1p4ssw0rd.!` Gràcies a la definició d'`Example5.USERNAME_TOKEN`, generem el testimoni d'usuari reemplaçant aquests valors, però també cal un valor per al *nonce* i un altre per a indicar quan es crea el testimoni i, finalment, definir l'identificador de l'element `<wsse:UsernameToken>`:

```
public static String getUsernameToken(String username,
String password) {
   ...
   Calendar cal = Calendar.getInstance(TimeZone.getDefault());
   SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
   SimpleDateFormat sdfTime = new SimpleDateFormat("HH:mm:ss");
   ...
   res = res.replaceAll("%%NONCE%%", getNonce());
   res = res.replaceAll("%%USERTOKENID%%",
   new BigInteger(130, new SecureRandom()).toString(32));
   res = res.replaceAll("%%CREATED%%", sdf.format(cal.getTime()) + "T" +
   sdfTime.format(cal.getTime()) + "Z");
   ...
}
```

Una vegada tenim el testimoni d'usuari representat en un objecte `org.w3c.dom.Element`, l'incloem dins de la capçalera SOAP:

```
SOAPEnvelope envelope = soapPart.getEnvelope();
SOAPHeader header = envelope.getHeader();
```

```
Node headerNode = header.getOwnerDocument().importNode(
   headerElem, true);
header.appendChild(headerNode);
```

D'aquesta manera, ja tenim un missatge SOAP a punt per a ser transmès a un servei web:

```
SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
SOAPConnection sc = scf.createConnection();
SOAPMessage reply = sc.call(message, "http://www.prise.es/service");
```

3.2.3. Testimoni XML

Un dels testimonis XML més populars és l'asserció SAML, de manera que descriurem aquest tipus de testimonis mitjançant els representats en aquesta tecnologia.

L'asserció SAML conté una sèrie d'afirmacions sobre un subjecte, i això pot ser validat de tres maneres diferents:

1) Holder of key. Confirmació que el subjecte era el propietari de la clau utilitzada en la generació de la signatura digital de l'asserció.

2) Sender vouchers. L'emissor de l'asserció SAML respon per la verificació de la identitat del subjecte.

3) Bearer. No s'aporta cap informació sobre la verificació de la identitat de l'usuari, de manera que depèn de la confiança establerta en l'entitat que ha generat l'asserció.

Exemple

En aquest subapartat analitzarem un exemple de transmissió de missatge SOAP incloent-hi un testimoni SAML amb verificació d'usuari *bearer*:

```
import com.sun.xml.wss.saml.*;
import java.io.StringReader;
import java.util.*;
import javax.xml.parsers.*;
import javax.xml.soap.*;
import org.w3c.dg.*;
import org.xml.sax.InputSource;

public class Example6 {

public static Assertion getSamlToken() throws Exception {
SAMLAssertionFactory af = SAMLAssertionFactory.newInstance(
SAMLAssertionFactory.SAML2_0);
String assertionId =
"a144i8f3&#8722;adad&#8722;594a&#8722;9649&#8722;924517abe933";
NameID nameId = af.createNameID("cr01", null, null);
GregorianCalendar issuerInst = new GregorianCalendar();
Subject subject = af.createSubject(nameId,
af.createSubjectConfirmation(null,
"urn:oasis:names:tc:SAML:2.0:cm:bearer"));
List<Object> statements = new<Object>LinkedList ();
List attributes = new LinkedList();
String nameAttr = "sn1";
List<String> valuesAttr = new<String>LinkedList ();

valuesAttr.add("Rodriguez");
attributes.add( af.createAttribute(nameAttr, valuesAttr) );
```

```
AttributeStatement attrSt = af.createAttributeStatement(attributes);
statements.add(attrSt);

AuthnContext ac = af.createAuthnContext(
"urn:oasis:names:tc:SAML:2.0:ac:classes:Password", null);
AuthnStatement as = af.createAuthnStatement(null, null, ac,
null, null);
statements.add(as);

Assertion assertion = af.createAssertion(assertionId, nameId,
issuerInst, null, null, subject, statements);
return assertion;
}

public static Element getElementFromString(String s) throws
Exception {
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
DocumentBuilder builder = factory.newDocumentBuilder();

InputSource is = new InputSource(new StringReader(s));
Document d = builder.parse(is);

return d.getDocumentElement();
}

public static void main(String[] args) throws Exception {
String reqMsg = "<requestMsg/>";
Element req = getElementFromString(reqMsg);
Assertion assertion = getSamlToken();
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
Element assertionElem = assertion.toElement(doc);
Element headerElem =
doc.createElementNS("http://docs.oasis-open.org/wss/2004/01/oasis
-200401-wss-wssecurity-secext-1.0.xsd", "Security");
headerElem.setPrefix("wsse");
headerElem.appendChild(assertionElem);

MessageFactory messageFactory = MessageFactory.newInstance();
SOAPMessage message = messageFactory.createMessage();
SOAPPart soapPart = message.getSOAPPart();
SOAPEnvelope envelope = soapPart.getEnvelope();
SOAPHeader header = envelope.getHeader();
org.w3c.dom.Node headerNode = header.getOwnerDocument().importNode
(headerElem, true);
header.appendChild(headerNode);
SOAPBody body = envelope.getBody();
body.addDocument(req.getOwnerDocument());
message.saveChanges();

SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
SOAPConnection sc = scf.createConnection();
SOAPMessage reply = sc.call(message, "http://www.prise.es/service");
}
}
```

Com veiem en l'exemple, el primer pas és crear l'asserció SAML, que obtenim en cridar en el nostre mètode principal a `getSamlToken()`, que retorna un objecte `com.sun.xml.wss.saml.Assertion`.

Aquest objecte és capaç de generar una representació de l'asserció SAML en un objecte `org.w3c.dom.Element`, per la qual cosa la manera en què generem la capçalera SOAP és diferent en aquest exemple:

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
factory.setNamespaceAware(true);
```

```
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
Element assertionElem = assertion.toElement(doc);
Element headerElem =
doc.createElementNS("http://docs.oasis-open.org/wss/2004/01/oasis-
 200401-wss-wssecurity-secext-1.0.xsd", "Security");
headerElem.setPrefix("wsse");
headerElem.appendChild(assertionElem);
```

Aquesta asserció SAML, però, viatja sense cap mena de verificació criptogràfica. Per això, podem signar l'asserció fàcilment incloent-hi aquestes ordres:

```
PublicKey pubKey = getPublicKey();
PrivateKey privKey = getPrivateKey();
Element assertionElem = assertion.sign(pubKey, privKey);
Document doc = assertionElem.getOwnerDocument();
Element headerElem =
doc.createElementNS("http://docs.oasis-open.org/wss/2004/01/oasis
-200401-wss-wssecurity-secext-1.0.xsd", "Security");
headerElem.setPrefix("wsse");
headerElem.appendChild(assertionElem);
```

Els mètodes `getPublicKey()` i `getPrivateKey()` obtenen una clau pública i privada, respectivament.

3.3. OpenID

OpenID és un estàndard d'autenticació d'usuaris que, des d'un punt de vista descentralitzat, permet accedir a recursos que tenen desplegat un control d'accés. A més, aquests usuaris poden accedir a diferents recursos amb una mateixa identitat digital, i proveir d'aquesta manera un sistema de *single sign-on*. El servei en què l'usuari s'autentica es diu *proveïdor d'OpenID* o *OpenID provider*, mentre que el recurs protegit és conegut com a *Relaying Party*.

La diferència principal amb els sistemes més comuns de federació d'identitat digital és que OpenID no proporciona un únic punt central per a autenticar l'usuari, sinó que hi ha nombrosos proveïdors d'OpenID desplegats a Internet que poden ser utilitzats per a accedir a qualsevol recurs protegit per aquesta tecnologia.

Afortunadament, tenim una àmplia col·lecció de biblioteques obertes per a treballar amb OpenID en gairebé totes les plataformes, tant en PHP com en Python i Javao.NET. Per als exemples d'aquesta secció, utilitzarem la biblioteca basada en PHP desenvolupada per Janrain.

3.3.1. Protocol d'autenticació

El protocol d'autenticació d'OpenID disponible en la versió 2 es basa en l'enviament de missatges HTTP utilitzant tant els mètodes GET com POST.

El flux de missatges que es produeixen a l'hora d'accedir un usuari a un Relaying Party és el següent:

1) L'usuari indica al Relaying Party quin és el seu identificador OpenID, anomenat també *user-supplied identifier*, mitjançant el seu navegador o agent. Per regla general, s'estableix aquest valor mitjançant un formulari HTML, que ha de tenir com a nom `openid_identifier`. D'aquesta manera, el navegador pot identificar clarament si hi ha la possibilitat d'autenticar-se en el Relaying Party mitjançant OpenID.

2) Una vegada el Relaying Party normalitza l'identificador OpenID, que pot ser un XRI o un URI, fa el procés de descobriment o *discovery*. D'aquesta manera, obté informació sobre l'URL del proveïdor d'OpenID de l'usuari.

3) Opcionalment, el proveïdor d'OpenID i el Relaying Party estableixen una associació que resulta en una clau secreta compartida per totes dues parts. D'aquesta manera, s'elimina la necessitat de verificar les signatures digitals cada vegada que se sol·licita l'autenticació de l'usuari.

4) El Relaying Party reencamina l'usuari mitjançant el seu navegador o agent al seu proveïdor d'OpenID amb un missatge de petició d'autenticació.

5) L'usuari s'autentica contra el seu proveïdor d'OpenID. Les qüestions relatives al procés de l'autenticació estan fora de l'àmbit del protocol, que queda delegat a les decisions dels responsables d'aquest proveïdor.

6) El proveïdor d'OpenID reencamina l'usuari, per mitjà del navegador o agent, al Relaying Party que va sol·licitar aquesta petició d'autenticació, i indica si l'autenticació va ser correcta o, en cas contrari, no va ser possible.

7) El Relaying Party verifica la informació que rep del proveïdor d'OpenID mitjançant el navegador o agent. Aquesta verificació implica comprovar els valors del missatge que rep, i també la signatura digital d'aquest missatge, sia mitjançant la clau compartida obtinguda en el pas 3 o enviant una petició directament al proveïdor d'OpenID de l'usuari.

3.3.2. Exemple de Relaying Party

Complementar, o fins i tot reemplaçar, la típica protecció amb una parella usuari i contrasenya amb accés mitjançant OpenID és realment senzill. En aquest exemple, aprendrem a programar en PHP el sistema necessari per a incloure autenticació mitjançant OpenID en les nostres aplicacions.

Basant-nos en l'exemple de consum de la biblioteca PHP de Janrain, que és al seu directori `php-openid/examples/consumer/`, generem l'estructura següent d'una aplicació web:

```
Auth/
common.php
index.php
```

```
try_auth.php
openid_response.php
```

Aquest conjunt de fitxers i directoris el podem dividir en tres grups:

1) Biblioteca per a treballar amb la tecnologia d'OpenID. En aquest grup hi ha el directori `Auth` i el fitxer `common.php`.

2) Inici de la sol·licitud d'autenticació OpenID, que correspon als fitxers `index.php` i `try_auth.php`.

3) Processament de la resposta d'autenticació, que correspon al fitxer `openid_response.php`.

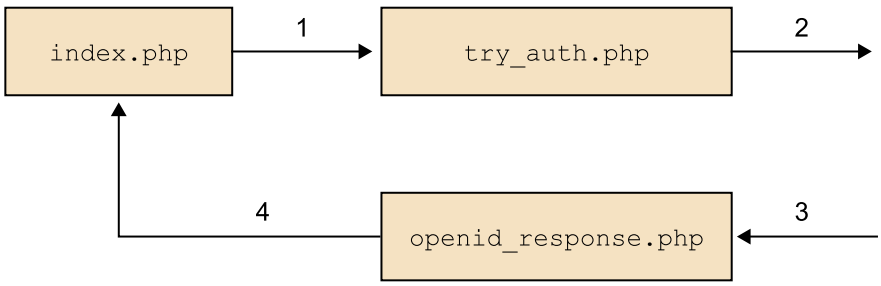
El directori `Auth` és la biblioteca per a treballar amb peticions cap a proveïdors d'OpenID i està inclòs dins de la distribució del programari de Janrain, mentre que el fitxer `common.php` és pràcticament igual al distribuït en aquesta biblioteca, tret que hàgim de modificar la funció `getReturnTo`:

```
function getReturnTo($file) {
return sprintf("%s://%s:%s%s/".$file,
getScheme(), $_SERVER['SERVER_NAME'],
$_SERVER['SERVER_PORT'],
dirname($_SERVER['PHP_SELF']));
}
```

D'aquesta manera, podem indicar quin fitxer volem usar per a la resposta del proveïdor d'OpenID.

La resta dels fitxers de l'exemple defineixen el flux de treball del nostre, tal com mostra la figura:

Diagrama de relació entre parts



El primer pas a l'hora d'incloure autenticació OpenID en el nostre recurs és incloure un formulari perquè l'usuari indiqui quin és el seu proveïdor d'OpenID, com veiem en l'exemple següent:

```
<html>
      <head>
```

```
            <title>Relaying Party OpenID Example</title>
      </head>
      <body>
<?php
session_start():
if (array_key_exists("logged", $_SESSION)) {
?>
<p>Autenticació correcta. El teu identificador OpenID és:</p>
<p><?php echo $_SESSION["logged"]; ?></p>
<?php
)
else {
      if (array_key_exists("openid_error_msg", $_REQUEST)) {
?>
<p>Error en l'autenticació OpenID:</p>
<p><?php echo $_REQUEST["openid_error_msg"]; ?></p>
<?php
                  }
?>
<p>S'ha d'autenticar mitjançant OpenID.</p>
      <form method="get" action="try_auth.php">
            URL identitat OpenID:
            <input type="hidden" name="action" value="verify" />
            <input type="text" name="openid_identifier" value="" />
            Input type="submit" value="Acceptar" />
      </form>
<?php
}
?>
      </body>
</html>
```

Com veiem en el codi, el primer que es fa és comprovar si hi ha algun valor de sessió per a la clau `logged`. En el processament de la resposta d'autenticació d'OpenID, guardem per a aquesta clau, en la sessió, l'identificador digital de l'usuari que ens retorna el seu proveïdor d'OpenID. Si és així, mostra en el navegador aquest identificador.

En cas contrari, vol dir que no està autenticat en el sistema, de manera que se li mostra el formulari en què ha d'indicar el seu proveïdor d'OpenID, que fa una petició HTTP amb mètode GET al fitxer `try_auth.php`. El paràmetre `openid_identifier` conté aquest valor de proveïdor d'OpenID, mentre que el camp `action` té el valor `verify`, que indica la biblioteca d'OpenID en la qual es vol iniciar el procés d'autenticació amb aquest proveïdor.

A més, si quan el fitxer `finish_auth.php` processa la resposta d'autenticació incorre en algun error, aquest fitxer ho indica amb el paràmetre `openid_error_msg`.

El fitxer `try_auth.php` conté la lògica que processa el valor introduït per l'usuari per a indicar el seu proveïdor d'OpenID i, si aquest proveïdor té un format correcte, inicia el procés d'autenticació.

```
<?php
require_once "common.php";
session_start();

$returnFile = 'openid_response.php';
$openid = $_GET['openid_identifier'];

$consumer = getConsumer();
$auth_request = $consumer->begin($openid);

if (!$auth_request) {
$msg_error = "Authentication error; not a valid OpenID.";
header("Location:            ".$returnFile."?openid_msg_error="
. base64_encode($msg_error));
exit();
}

if ($auth_request->shouldSendRedirect()) {
$redirect_url =             $auth_request->redirectURL(getTrustRoot(),
getReturnTo($returnFile));
if (Auth_OpenID::isFailure($redirect_url)) {
$msg_error = "Could not redirect to server: ".
$redirect_url->message;
header("Location:             ".$returnFile."?openid_msg_error=".
base64_encode($msg_error));
exit();
} else {
// Send redirect.
header("Location: ".$redirect_url);
exit();
}
} else {
$form_id = 'openid_message';
$form_html = $auth_request->htmlMarkup(getTrustRoot(),
getReturnTo($returnFile),
false,
array('id' => $form_id));
if (Auth_OpenID::isFailure($form_html)) {
$msg_error = "Could not redirect to server: ".
```

```
$form_html->message;
header("Location:             ".$returnFile."?openid_msg_error=".
 base64_encode($msg_error));
exit();
} else {
echo $form_html;
}
}
?>
```

Com s'observa en el codi, aquest fitxer processa el valor del camp `openid_identifier` i el normalitza, amb l'objectiu de tenir un URI complet al qual pugui fer la petició. Per a fer aquesta petició, primer comprova si el proveïdor d'OpenID de l'usuari funciona amb OpenID v2 o no, per a mostrar un formulari que s'executa automàticament, o bé, amb OpenID v1, fa una petició HTTP amb mètode GET. En aquesta petició hem d'indicar que volem que la resposta de la sol·licitud es processi en el fitxer `openid_response.php` del nostre projecte web.

```
<?php
require_once "common.php";
session_start();
function escape($thing) {
return htmlentities($thing);
}

function OpenID_finish_auth_success($openid_identity) {
session_start();
$_SESSION["logged"] = $openid_identity;
header ("Location: index.php");
exit;
}

function OpenID_finish_auth_error($return_to, $msg) {
header ("Location: index.php?openid_error_msg=".$msg);
exit;
}
$responseFile = 'openid_response.php';

$consumer = getConsumer();

// Complete the authentication process using the server's response.
$return_to = getReturnTo($responseFile);
$response = $consumer->complete($return_to);

// Check the response status.
if ($response->status == Auth_OpenID_CANCELL) {
```

```
// This means the authentication was cancelled.
$msg = 'Verification cancelled.';
OpenID_finish_auth_error($return_to,$msg);
} else if ($response->status == Auth_OpenID_FAILURE) {
// Authentication failed; display the error message.
$msg = "OpenID authentication failed: " . $response->message;
OpenID_finish_auth_error($return_to,$msg);
} else if ($response->status == Auth_OpenID_SUCCESS) {
// This means the authentication succeeded.
$openid = $response->getDisplayIdentifier();
$esc_identity = escape($openid);
OpenID_finish_auth_success($esc_identity);
}
?>
```

El primer pas per a processar la resposta d'autenticació és especificar que en la petició es va indicar que es volia la resposta en el fitxer `openid_response.php`, requisit de la biblioteca de Janrain per a comprovar que la nostra aplicació web espera la resposta.

Una vegada processada la resposta, tenim el resultat de l'autenticació en l'atribut `status` de l'objecte `$response`:

- `Auth_OpenID_CANCELL`: l'autenticació va ser cancel·lada per l'usuari en el proveïdor d'OpenID.

- `Auth_OpenID_FAILURE`: l'autenticació no va ser correcta, de manera que es reencamina l'usuari a `index.php` i s'indica en el paràmetre `openid_error_msg` quin va ser el motiu de la fallada.

- `Auth_OpenID_SUCCESS`: l'autenticació va ser correcta, de manera que es genera un context de seguretat en l'aplicació web, mitjançant la funció `OpenID_finish_auth_success`, i reencamina l'usuari a `index.php`. Un context de seguretat és un context web comú però amb un vincle i unes variables associades al domini d'identitat en què s'ha autenticat.

3.4. OAuth

Open Authorization (OAuth) és un estàndard obert que permet incloure seguretat en l'autenticació per a aplicacions web i d'escriptori.

OAuth implementa la manera de protegir no solament un recurs web, sinó també elements d'altres aplicacions, com són les d'escriptori o les mòbils.

La diferència entre OAuth i altres sistemes federats és que aquests últims ofereixen una identitat única per a accedir a molts recursos, mentre que OAuth permet a altres aplicacions l'accés a un recurs, però sense donar cap mena d'informació de caràcter personal. Això afavoreix les aplicacions web a oferir els seus serveis sense forçar els usuaris a exposar les seves contrasenyes o altres credencials.

L'estructura d'autorització i autenticació mitjançant OAuth es basa en diversos elements bàsics:

- **Proveïdor de servei** o ***service provider***. Aplicació web en què hi ha els recursos que estan protegits mitjançant OAuth i que s'encarrega de denegar o permetre l'accés a aquests recursos.

- **Usuari** o ***user***. Individu que té un compte en l'aplicació web proveïdora del servei.

- **Consumidor** o ***consumer***. Aplicació web que utilitza OAuth per a accedir al proveïdor de servei en nom de l'usuari.

- **Recurs protegit** o ***protected resource***. Dades controlades pel proveïdor de servei amb OAuth a les quals podrà accedir, mitjançant autenticació, el consumidor.

Per a garantir la seguretat en donar accés als recursos, OAuth defineix un mètode per a validar l'autenticitat de les peticions HTTP. Aquest mètode s'anomena *signing request* i intenta solucionar els aspectes següents:

- Les dades enviades amb les peticions s'han d'enviar encriptades per a evitar que una tercera persona les intercepti i en faci un ús il·lícit.

- Hi ha d'haver un mecanisme que associï una petició amb unes credencials concretes, ja que, si no es fa així, es podrien utilitzar unes credencials correctes en qualsevol petició que s'enviés.

- S'ha de permetre interactuar amb dos tipus de credencials: les del consumidor, que garanteixen que aquest consumidor està autoritzat prèviament, i les de l'usuari, que el certifiquen com a usuari del proveïdor del servei.

Segons el grau de seguretat del protocol utilitzat en la comunicació, el mètode varia. Si va sobre HTTPS, se segueix el mètode PLAINTEXT, que delega la major part de la seguretat en la capa HTTPS. Quan va sobre HTTP, el mètode de seguretat ha de ser molt més elaborat i és el que comentarem en aquest subapartat.

Per a acreditar el consumidor, s'utilitzen la clau del consumidor o *consumer key* i el consumidor secret o *consumer secret*, que identifiquen el consumidor davant el proveïdor de servei.

Per a identificar l'usuari, disposem d'un testimoni i del testimoni secret o *token secret*. Aquests elements representen l'usuari, però difereixen del nom d'usuari i contrasenya originals en el proveïdor de servei. Això permet al proveïdor de servei i a l'usuari tenir més control i seguretat, i a més dóna accés al consumidor i permet a un usuari revocar un testimoni sense haver de canviar contrasenyes en altres aplicacions.

Com a mètode per a garantir la integritat, OAuth utilitza signatures digitals en comptes d'enviar les credencials completes, i es verifica així que el contingut de la petició no s'ha modificat en el camí entre una aplicació i una altra. La garantia de confiança depèn de l'algorisme de signatura que s'utilitzi i de la manera d'aplicar aquest algorisme.

Com que aquesta signatura electrònica no verifica la identitat del qui envia el missatge, utilitzem la signatura combinada amb el secret compartit. Per a això, cal que tots dos elements acceptin un secret que només saben ells.

Per a no permetre reenviaments per terceres persones, OAuth incorpora dos elements: un número identificador per cada petició que el consumidor enviï al proveïdor de servei i una marca de temps que permet als proveïdors de servei mantenir els identificadors durant un temps limitat.

Hi ha tres mètodes de signatura diferents que engloben les característiques comentades en aquest subapartat:

- **PLAINTEXT**. És el més simple de tots tres i només es recomana utilitzar sobre HTTPS.

- **HMAC-SHA1**. Combina HMAC, que ofereix secret compartit simètric i SHA1 com a algorisme de *hash*.

- **RSA-SHA1**. És el més complex de tots tres i combina RSA com a mecanisme de seguretat de les claus i SHA1 com a algorisme de *hash*.

3.4.1. Protocol d'autenticació

L'autenticació mitjançant OAuth és el procés mitjançant el qual els usuaris concedeixen accés als seus recursos protegits sense compartir les seves credencials amb el consumidor.

En comptes d'utilitzar les credencials de l'usuari en les peticions a recursos protegits, OAuth utilitza els testimonis generats pel proveïdor de servei que anomenem *request token* i *access token*:

- ***Request token***. Utilitzat pel consumidor per a demanar a l'usuari accés als recursos protegits. Una vegada el *request token* és autoritzat per l'usuari, s'intercanvia per un *access token*, que s'ha d'utilitzar només una vegada i no es pot usar per a una altra comesa.

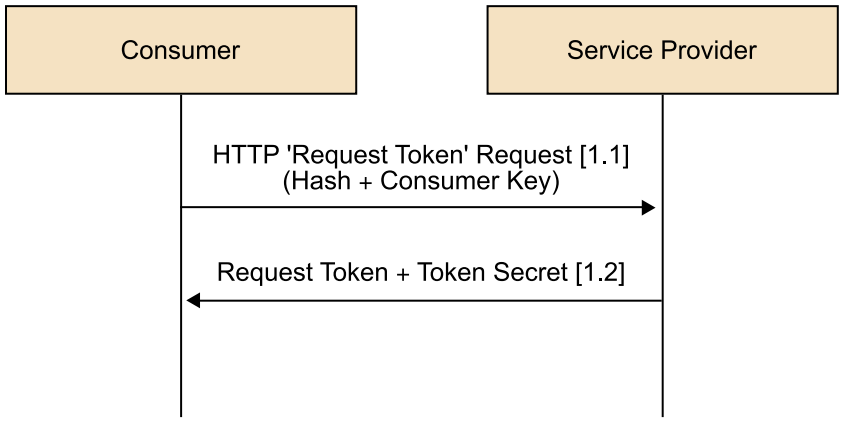
- ***Access token***. Utilitzat pel consumidor per a accedir als recursos protegits en nom de l'usuari. Pot limitar més d'un recurs i tenir un temps de vida limitat. És l'únic testimoni que es pot usar per a accedir als recursos protegits, i és susceptible de ser revocat.

L'autenticació d'OAuth es fa en tres passos:

Pas 1. El consumidor obté un *request token* sense autoritzar enviant una petició HTTP a l'URL disposat per a fer-ho en el proveïdor de servei. La petició l'ha d'haver signat el consumidor i ha de contenir la clau del consumidor. Una vegada rebuda, el proveïdor de servei verifica la signatura i la clau del consumidor i, si són correctes, genera un *request token* i un *token secret* i els retorna al consumidor.

En el diagrama de flux de la figura, veiem com es comuniquen el consumidor i el proveïdor de servei en aquesta primera fase:

Flux de missatges OAuth en una primera fase



La petició HTTP ha de ser POST i té el format següent:

```
GET /directori/recursos?nom=foto1.jpg
HTTP/1.1
Host: http://serviceprovider.com
Content-Type: application/atom+xml
Authorization:
OAuth oauth_token="1%2Fab3cd9j4ks73hf7g",
```

```
oauth_signature_method="RSA-SHA1",
oauth_signature="wOJIO9AvZbTSMK%2FPY%3D...",
oauth_consumer_key="exemple.com",
oauth_timestamp="137131200",
oauth_nonce="4572616i48616d6d",
oauth_version="1.0"
```

La resposta conté almenys els paràmetres següents:

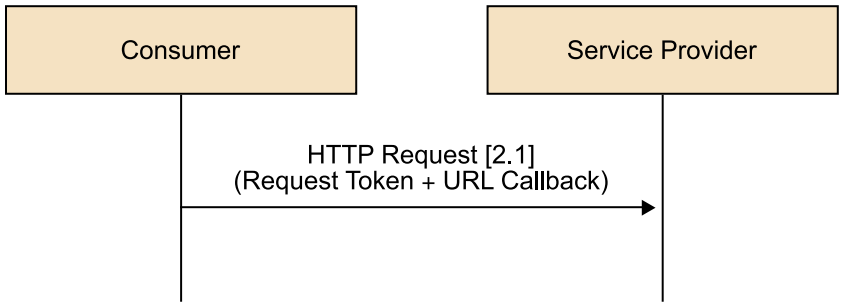
```
oauth_token: 1%2Fab3cd9j4ks73hf7g
oauth_token_secret: 47ba47i0048b7f2105db67df18ffd24bd072688a
```

Com veiem, l'`oauth_token` conté el mateix valor que el que tenia el paràmetre `oauth_token` en la petició, ja que representa el *request token*.

Pas 2. El consumidor reencamina l'usuari a l'URL que el proveïdor de servei ha disposat per a l'autenticació, i envia al seu torn una petició HTTP amb el *request token* i l'URL al qual ha de retornar l'usuari quan s'ha acabat el procés. En aquest punt, l'usuari s'autentica en el proveïdor de servei i estableix la política d'accés que vulgui per als recursos protegits. Una vegada ha acabat, el proveïdor de servei retorna l'usuari al consumidor tal com es va establir en la petició.

En el diagrama de la figura observem el flux de peticions que es fan en aquesta segona etapa:

Flux de missatges OAuth en la segona etapa



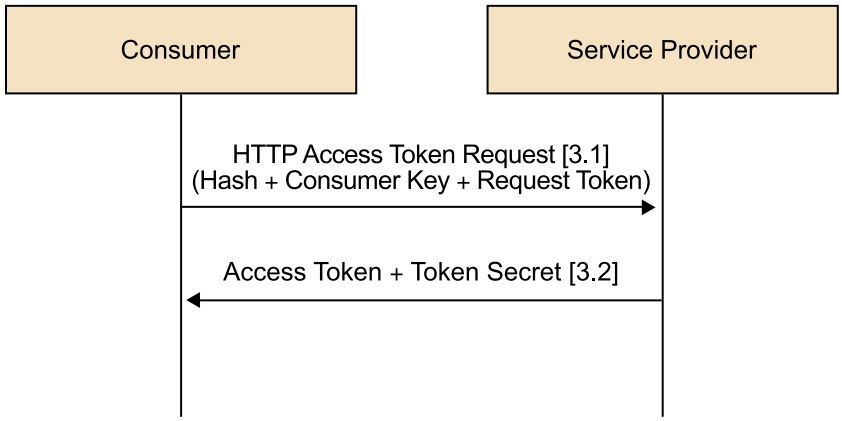
El *request token* és el generat anteriorment, i l'URL de crida de retorn o *callback* és l'URL al qual ha de reencaminar el proveïdor de servei l'usuari una vegada tractats els permisos d'accés dels recursos protegits.

Pas 3. Una vegada autoritzat, el consumidor intercanvia el *request token* per un *access token* que sigui capaç d'accedir als recursos protegits. Per a fer-ho, el consumidor fa una petició HTTP d'un *access token* a l'URL especificat pel proveïdor de servei. Aquesta petició ha d'estar signada mitjançant el mètode *signing request* especificat més amunt i ha de contenir la clau del consumidor, el *request token* i la signatura, entre altres paràmetres.

Per a donar accés a un recurs, el proveïdor de servei s'ha d'assegurar que es verifiqui la signatura de la petició, que aquest *request token* encara no ha estat intercanviat i que la clau del consumidor coincideixi amb la del *request token* especificat. Si es compleix tot això, el proveïdor de servei genera un *access token* i un *token secret* –que és un secret que utilitza el consumidor i que l'acredita com a propietari d'un testimoni determinat–, i els retorna al consumidor. Aquestes dades són emmagatzemades i són utilitzades per a signar les peticions dels recursos protegits.

En el diagrama de la figura veiem el flux de missatges que es fa en aquesta tercera etapa de la comunicació entre el consumidor i el proveïdor de servei:

Flux de missatges OAuth en la tercera etapa



La petició HTTP conté la signatura digital, la clau del consumidor i el *request token* que es vol intercanviar (que és similar a l'intercanviat en els passos anteriors).

La resposta del proveïdor de servei ha d'incloure els paràmetres següents:

```
oauth_token: 1%2Fab3cd9j4ks73hf7g
oauth_token_secret: 34ba34i0048bdse6505db67df165fd24b7672688a
```

El paràmetre `oauth_token` conté l'*access token* generat pel proveïdor de servei, mentre que el paràmetre `oauth_token_secret` acredita el consumidor identificant-lo.

3.4.2. Exemple d'un consumidor

A continuació, mostrem un exemple en què expliquem com s'ha d'implementar un consumidor per a interactuar amb un proveïdor de servei mitjançant OAuth.

Nota

La biblioteca utilitzada en aquest subapartat és l'`oauth-php`.

En primer lloc, s'ha de crear una base de dades que serveixi al consumidor com a mitjà d'emmagatzematge de la informació obtinguda amb els testimonis i les peticions HTTP. Per a fer-ho, hem d'utilitzar l'*script* PHP que hi ha

a `library/store/mysql/install.php` i que importa l'estructura de la base de dades que ens dóna la biblioteca i que és a `library/store/mysql/mysql.sql`.

Una vegada fet això, s'ha d'obtenir una instància que permeti accedir a la base de dades, cosa que farem de la manera següent:

```
$bd = OAuthStore::instance('tipusBD', $opcions);
```

En què `tipusBD` és el tipus de base de dades (per exemple, MySQL) i `$options`, una matriu o *array* amb els atributs servidor, contrasenya, nom d'usuari i base de dades.

En segon lloc, hem d'afegir el servidor a la instància de la base de dades per a interactuar-hi. Primer creem una matriu amb les dades del consumidor respecte a un proveïdor de servei:

```
$descserv = array( 'consumer_key' => 'Clau_Del_Consumidor_En_El_SP',
'consumer_secret' => 'Secret_del_Consumidor',
'server_uri' => 'http://sp_exemple/',
'signature_methods' => array('HMAC-SHA1','PLAINTEXT'),
'request_token_uri' => 'http://sp_exemple/request_token',
'authorize_uri' => 'http://sp_exemple/authorize',
'access_token_uri' => 'http://sp_exemple/access_token' );
```

Després declarem i inicialitzem un identificador de l'usuari els recursos del qual volem utilitzar, i emmagatzemem a la base de dades el servidor que tenim a la matriu `$descserv`.

```
$ident_user = 1;
$consumer_key = $bd->updateServer($descserv, $ident_user);
```

Per a tractar el procés d'autenticació, hem de declarar i inicialitzar l'URL de retorn que s'ha d'enviar al proveïdor de servei, al qual afegim la clau del consumidor codificada segons l'RFC 1738 i l'identificador de l'usuari corresponent.

```
$callback_uri = 'http://consumidor.com/callback?
consumer_key='.rawurlencode($consumer_key).'
&usr_id='.intval($ident_user);
```

Pas 1. El consumidor obté un *request token* sense autoritzar enviant una petició al proveïdor de servei. Això es duu a terme mitjançant l'ordre següent:

```
$token = OauthRequester::requestRequestToken($consumer_key,
 $ident_user);
```

Pas 2. El consumidor reencamina l'usuari a l'URL que el proveïdor de servei ha disposat per a l'autenticació en cas d'haver rebut un testimoni autoritzat en el pas 1, i envia, en reencaminar l'usuari, el *request token* i l'URL, que s'ha de retornar a l'usuari una vegada s'ha acabat el procés.

```
if (!empty($token['authorize_uri'])){
if (strpos($token['authorize_uri'], '?')) {
$uri = $token['authorize_uri'] . '&';
}else {
$uri = $token['authorize_uri'] . '?';
}
$uri .= 'oauth_token='.rawurlencode($token['token']).'
&oauth_callback='.rawurlencode($callback_uri);
}else {
$uri = $callback_uri . '&oauth_token='.rawurlencode($token['token']);
}
header('Location: '.$uri);
exit();
```

Una vegada l'usuari ha estat autoritzat i el proveïdor de servei el reencamina a l'adreça de crida de retorn proporcionada, se sol·licita l'intercanvi del *request token* per l'*access token*.

```
$consumer_key = $_GET['consumer_key'];
$oauth_token = $_GET['oauth_token'];
$ident_user = $_GET['usr_id'];
OAuthRequester::requestAccessToken($consumer_key, $oauth_token,
$ident_user);
```

Després de fer aquests passos, ja estem a punt per a demanar l'accés als recursos protegits en qüestió mitjançant l'ordre:

```
$request = new OAuthRequester($uri_request, 'GET', $parametres);
```

En què `$uri_request` és l'URI del proveïdor de servei al qual accedirem (que ens ha estat facilitar en registrar el servidor); el segon paràmetre pot ser tant GET com POST, i `$parametres` és una matriu en què s'afegeixen els paràmetres requerits pel proveïdor de servei per a funcionar.

Finalment, només ens falta fer la petició mitjançant l'ordre de codi següent:

```
$result = $req->doRequest($user_id);
```

El resultat és una matriu amb els paràmetres `code` (un enter), `headers` (una matriu) i `body` (una cadena), que depenen del proveïdor de servei a què s'accedeix.

3.5. XACML

Quan parlem de gestionar centenars o milers de sistemes d'informació, la creació i la distribució de polítiques de seguretat no és una tasca fàcil. Molts d'aquests sistemes són diferents entre si i tenen una manera d'implementar les polítiques que disten molt els uns dels altres. Qualsevol canvi, encara que sigui mínim, implica que les desenes d'administradors que hi ha entenguin la política correctament i modifiquin tots i cadascun dels sistemes esmentats en la mateixa ordre. En aquest context, hi ha moltes probabilitats de cometre un error.

Amb aquest objectiu, l'organització OASIS va crear XACML[9], un llenguatge basat en XML per a emmagatzemar i distribuir polítiques de control d'accés. Encara no és un estàndard acceptat àmpliament i té altres competidors com PERMIS, però té un ecosistema de desenvolupament i suport interessants, com és el projecte HERAS.

(9) *XACML* és l'abreviació d'*eXtensible Access Control Markup Language*.

Enllaç d'interès

El projecte PERMIS és un motor de decisió RBAC basat en polítiques. Consulteu-lo a l'adreça web següent: http://www.openpermis.org.

HERAS

HERAS és un *framework* de suport a XACML 2.0 i per a la seguretat d'aplicacions web. Per a saber més coses sobre HERAS visiteu l'adreça web següent: http://www.herasaf.org.

Hi ha moltes complementarietats amb SAML: tant l'un com l'altre són llenguatges de pregunta-resposta, basats en XML, amb els mateixos termes (subjecte, objecte, etc.), però mentre que SAML ofereix un llenguatge per a traspassar la identitat i el control d'accés, XACML dicta què s'ha de fer amb la informació. Es pot recolzar, per això, en molts atributs de l'usuari o del context, de manera que *a priori* pot semblar un llenguatge més complicat.

Polítiques

Vegem alguns exemples de polítiques:

- Només es pot accedir a aquest conjunt de documents de les nou del matí a les cinc de la tarda.
- Únicament hi tenen accés de modificació els membres del departament de recursos humans, mentre que la resta únicament els pot llegir.
- Si hi accedim amb HTTPS hi tenim accés; d'una altra manera no.
- Si hi accedim amb HTTPS i pertanyem al departament de recursos humans hi tenim accés d'escriptura; d'una altra manera, només hi tenim accés de lectura.

Normalment aquestes polítiques es distribueixen en els *policy decision points* (PDP), punts dins d'una arquitectura d'identitat que són els encarregats de prendre decisions sobre la identitat i el control d'accés. Dins d'una estructura real d'identitat hi pot haver molts punts d'aquest tipus.

Els components d'XACML són els següents:

- **Policy**. Representa una política única de control d'accés, expressada per una sèrie de regles. És la base per a l'autorització de la decisió.

- **PolicySet**. Aglutina una sèrie de polítiques o conjunt de polítiques i procediments de la manera de combinar el resultat de les seves avaluacions. Serveix principalment per a combinar diferents polítiques en una d'única.

- **Rule**. Expressió booleana que es pot avaluar de manera aïllada. Es pot reutilitzar per a diverses polítiques.

- **Target**. Defineix un conjunt d'objectes, subjectes i accions als quals aplica una determinada regla.

- **Obligation**. Operació definida en una política (o conjunt de polítiques) que s'ha de dur a terme al costat de l'avaluació de les decisions d'autorització.

- **Condition**. Expressió que avalua `True`, `False` o `Indeterminate`.

- **Effect**. Conseqüència d'una regla satisfeta, permesa o denegada.

Exemple

Imaginem-nos que tenim una política que dicta això:

"Per a accedir a un servidor que es diu *SampleServer*, l'usuari s'ha d'haver identificat i autenticat (inici de sessió o *login*) contra el sistema i ho ha de fer entre les nou del matí i les cinc de la tarda."

La forma de la política és la següent:

```
<Policy PolicyId="SamplePolicy" RuleCombiningAlgId="urn:oasis:names:
  tc: xacml:1.0: rule-combining-algorithm: first-applicable">

(POLICY PARAMETERS)

(RULE1)

(RULE2)

...

<! -- A final, "fall-through" Rule that always Denies -->
<Rule RuleId="FinalRule" Effect="Deny"/>

</Policy>
```

El *target* s'expressa d'aquesta manera:

```
<! -- This Policy only applies to requests on the Sample Server -->
<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
<ResourceMatch MatchId="urn:oasis:names: tc:xacml:1.0: function:
 string-equal">
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
 SampleServer</AttributeValue>
<ResourceAttributeDesignator
 DataType="http: //www. w3.org/2001/XMLSchema#string"AttributeId=
 "urn: oasis: names: tc:
 xacml:1. 0: resource: resource-id"/>
</ResourceMatch>
</Resources>
```

```
<Actions>
<AnyAction/>
</Actions>
</Target>
```

La regla que ha d'estar identificada i autenticada contra el sistema segueix el patró següent:

```
<! -- Rule to see if we should allow the Subject to login -->

<Rule RuleId="LoginRule" Effect="Permit">
<! -- Only use this Rule if the action is login -->
<Target>
<Subjects>
<AnySubject/>
</Subjects>
<Resources>
<AnyResource/>
</Resources>
<Actions>
<ActionMatch MatchId="urn:oasis:names: tc:xacml:1.0: function:
 string-equal">
<AttributeValue  DataType="http: //www. w3. org/2001/
 XMLSchema#string">login</AttributeValue>
<ActionAttributeDesignator  DataType="http://www.w3.org/2001/
 XMLSchema#string"
 AttributeId="ServerAction"/>
</ActionMatch>
</Actions>
</Target>
```

La condició que ha de ser entre les nou del matí i les cinc de la tarda s'indica de la manera següent:

```
<! -- Only allow logins from 9am to 5pm -->

<Condition FunctionId="urn: oasis: names: tc: xacml: 1.0: function:
 and">
<Apply FunctionId="urn: oasis: names: tc: xacml: 1.0: function:
 time-greater-than-or-equal">
<Apply FunctionId="urn: oasis: names: tc: xacml: 1.0: function:
time-one-and-only">
<EnvironmentAttributeSelector DataType="http://www.w3.org/2001/
 XMLSchema#time"
 AttributeId="urn:oasis: names: tc: xacml: 1. 0: environment:
 current-time"/>
</Apply>
<AttributeValue DataType="http: //www. w3. org/2001/XMLSchema#time">
 09:00:00</AttributeValue>
</Apply>
<Apply FunctionId="urn: oasis: names: tc: xacml: 1. 0: function:
 time-less-than-or-equal">
<Apply FunctionId="urn: oasis: names: tc: xacml: 1. 0: function:
 time-one-and-only">
<EnvironmentAttributeSelector DataType="http://www.w3. org/2001/
 XMLSchema#time"
 AttributeId="urn:oasis: names: tc: xacml: 1. 0: environment:
 current-time"/>
</Apply>
<AttributeValue DataType="http: //www.w3.org/2001/XMLSchema#time">
 17: 00:00</AttributeValue>
</Apply>
</Condition>
</Rule>
```

3.6. SPML

SPML són les sigles de *service provisioning markup language*, llenguatge basat en XML per a l'aprovisionament automàtic d'identitats en sistemes d'informació i la gestió d'una identitat al llarg del seu cicle de vida, des de la creació fins a la revocació, passant per la modificació.

S'identifiquen tres rols diferents:

- ***Requesting authority*** (RA): entitat que fa la petició d'aprovisionament.

- ***Provisioning service provider*** (PSP): programari capaç de respondre a peticions SPML.

- ***Provisioning service target*** (PST): proveïdor de les identitats. Aquest rol pot estar unit a PSP, però mentre PSP ha d'entendre SPML, PST no.

SPML constitueix un llenguatge de pregunta-resposta, com SAML. Per això cal que RA i PSP estableixin una relació de confiança, que pot ser representada també mitjançant SAML, introduint les assercions corresponents en la capçalera i les instruccions SPML en el *payload*.

Té bàsicament cinc operacions:

1) `<addRequest/>`. S'utilitza per a crear un compte.

2) `<modifyRequest/>`. Es fa servir per a actualitzar un compte.

3) `<deleteRequest/>`. S'utilitza per a demanar esborrar un compte.

4) `<searchRequest/>`. S'utilitza per a interrogar el PSP sobre comptes i les propietats d'aquests comptes.

5) `<schemaRequest/>`. Permet demanar a l'RA l'esquema d'aprovisionament d'un PSP.

Exemple

Com a exemple d'això, mostrem com és una petició de creació d'un compte:

```
<addRequest>
      <attributes>
            <attr name="objectclass">
                  <value>urn:...:SPML:interop:interopUser</value>
            </attr>
            <attr name="cn">
                  <value>Mario López</value>
            </attr>
            <attr name="mail">
                  <value>mlopez@micompania.com</value>
            </attr>
            <attr name="registrationTime">
                  <value>21-Jul-2011 12:00</value>
            </attr>
      </attributes>
</addRequest>
```

El PSP hi pot respondre de la manera següent:

```
<addResponse result = "urn:...:SPML:1:0#success">
            <identifier type="urn:...:SPML:1:0#EMailAddress">
                  <id>mlopez@micompania.com</id>
            </identifier>
            <attributes>
                  <attr name="mailBoxLimit">
                        <value>50MB</value>
                  </attr>
            </attributes>
</addResponse>
```

3.7. Altres tecnologies delegades de control d'accés

Durant la història dels sistemes d'informació, la seguretat ha generat un gran interès i ha donat lloc a diferents tècniques, mètodes, tecnologies i teories. Dins de la gestió de la identitat i l'accés, no ha estat diferent, i s'han proposat diferents sistemes en aquest àmbit.

No obstant això, hem de diferenciar els sistemes que hi ha en matèria de control d'accés, com són Kerberos o Radius, dels sistemes federats, que obeeixen necessàriament a una realitat diferent, amb molts dominis d'identitat o seguretat que poden pertànyer a diferents organitzacions, amb polítiques i acostaments a la gestió de les identitats dispars.

Tanmateix, és interessant que repassem alguna d'aquestes tecnologies amb l'objectiu de fer un exercici de diferenciació respecte al que vol dir realment la federació d'identitats.

3.7.1. Kerberos

Kerberos, possiblement, és el sistema de control d'accés més conegut i utilitzat; per exemple, és emprat pels actuals sistemes Windows i, durant anys, pels sistemes Unix.

Concretament, Kerberos és un sistema de gestió d'accés per a sistemes heterogenis, a més d'un sistema de *single sign-on* i de distribució de claus. Proveeix de capacitats d'autenticació i gestió d'accés una sèrie de principals, és a dir, els components que cal protegir dins del sistema.

El component més important de Kerberos és el Key Distribution Center (KDC), en el qual confien tots els elements i és l'encarregat de guardar els secrets que serviran per a dur a terme els processos de seguretat que gestiona Kerberos. A grans trets, quan algú vol accedir a un recurs, com per exemple la impressora, demana l'accés al KDC amb les seves credencials. El KDC mira la taula de capacitats de l'usuari i, si hi pot accedir, li dóna un tiquet perquè el presenti a la impressora en el moment de sol·licitar la impressió.

Encara que Kerberos gestiona diferents recursos, no maneja diferents dominis d'identitat, amb diferents ID, polítiques, estructures, etc., sinó que els posa en comú i fa abstracció de la manera com està fet cada repositori d'identitat. En aquest cas, els recursos que parlen del protocol de Kerberos estan supeditats a un únic domini d'identitat.

Normalment, se sol utilitzar dins d'una mateixa empresa encara que hi ha alguns fabricants que han intentat estendre el protocol per donar-hi més abast.

SESAME

Secure European System for Applications in a Multi-vendor Environment (SESAME), per exemple, és un nou acostament a Kerberos que vol corregir alguns aspectes d'aquest últim, però que encara no disposa dels elements que permetin constituir un sistema per a la federació d'identitats.

3.7.2. Radius

El protocol Radius[10] (o la versió posterior, Diameter) i després TACACS[11] permeten l'autenticació remota de persones i sistemes i els donen accés després a una sèrie de recursos, amb autenticació prèvia negociada amb protocols com PAP (protocol de verificació de contrasenya o *password authentication protocol*), CHAP (*protocol d'autenticació per repte-negociació* o *challenge-handshake authentication protocol*) o EAP (protocol d'autenticació escalable o *extensible authentication protocol*). Radius utilitzava connexions PPP (protocol punt a punt o *point to point protocol*) i més endavant TACACS va permetre connexions TCP.

Encara que permet l'accés a sistemes remots, no federa la identitat. Per a connectar diferents sistemes Radius (o TACACS), s'han d'utilitzar tantes credencials com sistemes vulguem utilitzar. En aquest sentit, es tracta d'un protocol d'accés remot però no de federació d'identitat.

(10) *Radius* és l'abreviació de *servei d'autenticació remota de crides d'usuaris* o *remote authentication dial in user service*.

(11) *TACACS* és la sigla de *sistema de control d'accés al controlador d'accés al terminal* o *terminal access controller access-control system*.

3.7.3. 802.1x

El 802.1x és un estàndard de l'Institut d'Enginyers Elèctrics i Electrònics (IEEE) per a l'autenticació de dispositius basada en ports en una xarxa commutada. En aquest protocol i en molts altres d'aquesta índole, permeten l'autenticació a dominis d'identitat diferents. No obstant això, el 802.1x no federa la identitat, i s'encarrega únicament de l'operació d'autenticació.

Resum

La identitat, com a concepte fonamental en les nostres vides, ha estat objecte de molts acostaments en el trasllat que se n'ha fet al món virtual. Hi ha hagut diferents aportacions que han intentat, més o menys globalment, solucionar diferents aspectes de la gestió de la identitat i l'accés als recursos.

En aquest mòdul hem exemplificat al començament alguns escenaris d'ús de la federació d'identitats, com a concepte real que soluciona algun dels problemes que hi ha en la interacció entre diferents dominis d'identitat o seguretat. A partir d'això, hem definit uns conceptes base que ens han permès unificar termes que no sempre són clars en la literatura.

Tot seguit, hem repassat els patrons de federació que hi ha i que formen el marc de relació bàsica entre les parts. Depenent d'aquest model de relació, pot ser que les tècniques, les polítiques, els mètodes i els estàndards difereixin, i també l'estratègia mateixa de l'empresa pel que fa a gestió de la identitat i de l'accés. D'aquesta manera, ens hem acostat als patrons *ad hoc*, al *hub and spoke* i a la Federation Network.

Després, malgrat la fragmentació que hi ha en el món de la federació d'identitats (o almenys en alguna de les seves operacions), hem revisat les iniciatives més importants que pretenen afrontar el problema amb l'objectiu de crear estàndards i especificacions per a resoldre els reptes plantejats en la gestió de la identitat i l'accés federat. Les més rellevants que hem descrit en aquest mòdul són les iniciatives de Microsoft/IBM, OASIS, Liberty Alliance i Internet2 amb Shibboleth.

Finalment, hem revisat les tecnologies i els protocols més importants del món de la federació d'identitats. SAML ha ocupat gran part del nostre discurs, ja que es tracta d'un protocol que, de mica en mica, s'imposa com a mitjà de transmissió de la identitat i del control d'accés i també per la gestió d'atributs que fa. Tot seguit, hem vist que la federació d'identitats es transporta més enllà del món de les persones, concretament al dels serveis web, i hem descrit l'estàndard WSS. Després, continuant la descripció de tecnologies, hem afrontat OpenID com a mètode popular distribuït per a la gestió d'identitats, OAuth per a l'autorització, XACML per a la definició i la transmissió de polítiques d'identitat i de gestió d'accés i SPML com a llenguatge per a l'aprovisionament de sistemes d'identitat. Finalment, hem completat aquest apartat tecnològic posant en context les tecnologies descrites amb d'altres que ja hi havia en el món dels sistemes d'informació com Kerberos.

El viatge en què ens hem embarcat al llarg d'aquest mòdul constitueix, en si mateix, un repte emocionant i d'evolució constant. L'explosió d'iniciatives i de diferents acostaments pot fer pensar en una incertesa futura, però es venç àmpliament per la solidesa d'algunes iniciatives i les possibilitats que tenen.

Activitats

1. Instal·leu OpenAM, que és un sistema de control d'accés de codi lliure, i intenteu federar-lo amb algun dels proveïdors de servei que hi ha d'OpenID, com Google o Yahoo.

2. Les identitats de les persones sempre han estat un bé molt preuat per les grans companyies. Després d'iniciatives com la de Microsoft Passport, Facebook Connect vol aprofitar la força d'una xarxa social de més de sis-cents milions d'usuaris per a vincular els processos d'autenticació de tercers.

Busqueu informació sobre Facebook Connect, classifiqueu-la en algun dels patrons que hem vist i feu algunes proves d'autenticació des del vostre propi web de proves.

3. Hi ha hagut moltes iniciatives de *single sign-on*: OpenSSO (ara OpenAM), JBoss SSO, Ubuntu SSO, etc. Busqueu les iniciatives que hi ha i agrupeu-les en els corrents que hem vist en el mòdul. S'assemblen més a les d'OASIS? O potser més a les de Liberty Alliance? Són propietàries?

Glossari

array *f* Vegeu **matriu**.

atribut *m* Característica d'una entitat.

base 64 *f* Sistema de codificació en el qual la base és de seixanta-quatre posicions, en lloc de la decimal habitual, que és de deu.

challenge-handshake authentication protocol *m* Vegeu **protocol d'autenticació per repte-negociació**.

CHAP *m* Vegeu **protocol d'autenticació per repte-negociació**.

codi d'autenticació de missatge basat en *hash* *m* Construcció, en criptografia, per a calcular un codi d'autenticació de missatge.
en challenge-handshake authentication protocol
sigla **CHAP**

EAP *m* Vegeu **protocol d'autenticació escalable**.

extensible authentication protocol *m* Vegeu **protocol d'autenticació escalable**.

extensible markup language *m* Vegeu **llenguatge d'etiquetatge escalable**.

extensible resource identifier *m* Vegeu **identificador de recursos escalables**.

hashed message authentication code *m* Vegeu **codi d'autenticació de missatge basat en *hash***.

HMAC *m* Vegeu **codi d'autenticació de missatge basat en *hash***.

HTTP *m* Vegeu **protocol de transferència d'hipertext**.

HTTP GET *m* Resposta HTTP en què els paràmetres van en l'URL.

HTTP POST *m* Resposta HTTP en què els paràmetres s'envien en el *payload* del paquet.

HTTPS *m* Sistema HTTP segur.

hypertext transfer protocol *m* Vegeu **protocol de transferència d'hipertext**.

identificador de recursos escalables *m* Sistema nou d'identificació a Internet, dissenyat específicament per a identitats digitals de domini encreuat.
en extensible authentication protocol
sigla **EAP**

identificador uniforme de recursos *m* Camí unívoc, dins d'un URL, a l'objecte referenciat.
en uniform resource identifier
sigla **URI**

IETF *m* Vegeu **Internet Engineering Task Force**.

inici de sessió *m* Procés d'autenticació per mitjà d'usuari i contrasenya, generalment.
en login

Internet Engineering Task Force *m* Organisme que desenvolupa i promou estàndards d'Internet.
sigla **IETF**

Java *m* Llenguatge de programació.

llenguatge d'etiquetatge escalable *m* Llenguatge de marcatge sobre la base d'un fitxer pla.
en extensible markup language
sigla **XML**

localitzador uniforme de recursos *m* Identificador global únic d'un recurs en el Web.
en uniform resource locator
sigla **URL**

login *m* Vegeu **inici de sessió**.

matriu *f* Col·lecció ordenada d'objectes.
en array

MySQL *f* Base de dades de codi lliure.

.NET *m* Llenguatge de programació.

PAP *m* Vegeu **protocol de verificació de contrasenya**.

password authentication protocol *m* Vegeu **protocol de verificació de contrasenya**.

petició de comentaris *f* Estàndard *de facto* col·laboratiu que emet l'IETF.
en request for comments
sigla **RFC**

petició-resposta *f* Model de programació interrogatiu.
en request-response

PHP *m* Llenguatge de programació.

point to point protocol *m* Vegeu **protocol punt a punt**.

PPP *m* Vegeu **protocol punt a punt**.

protocol d'accés a objecte simple *m* Protocol utilitzat per a cridar a objectes.
en simple object access protocol
sigla **SOAP**

protocol d'autenticació escalable *m* *Framework* d'autenticació utilitzat normalment en xarxes sense fil.
en extensible authentication protocol
sigla **EAP**

protocol d'autenticació per repte-negociació *m* Protocol que serveix per a autenticar dispositius mitjançant un protocol de repte-resposta.
en challenge-handshake authentication protocol
sigla **CHAP**

protocol de transferència d'hipertext *m* Protocol de nivell 4 per al transport de serveis en una pila IP.
en hypertext transfer protocol
sigla **HTPP**

protocol de verificació de contrasenya *m* Protocol per a autenticar-se contra un sistema remot.
en password authentication protocol
sigla **PAP**

protocol punt a punt *m* Protocol utilitzat per a la connexió a sistemes remots.
en point to point protocol
sigla **PPP**

Python *m* Llenguatge de programació.

request for comments *f* Vegeu **petició de comentaris**.

request-response *f* Vegeu **petició-resposta**.

RFC *f* Vegeu **petició de comentaris**.

servei web *m* Servei disponible per HTTP/HTTPS.

SHA-1 *m* Algorisme criptogràfic de *hash*.

simple object access protocol *m* Vegeu **protocol d'accés a objecte simple**.

SOAP *m* Vegeu **protocol d'accés a objecte simple**.

testimoni *m* Element que només pot tenir una entitat en un mateix moment i que és diferent dels altres testimonis, és a dir que és identificable.

thread *m* Fil d'execució que comparteix recursos computacionals amb el procés.

token *m* Vegeu **testimoni**.

uniform resource identifier *m* Vegeu **identificador uniforme de recursos**.

uniform resource locator *m* Vegeu **localitzador uniforme de recursos**.

URI *m* Vegeu **identificador uniforme de recursos**.

URL *m* Vegeu **localitzador uniforme de recursos**.

web service *m* Vegeu **servei web**.

XML *m* Vegeu **llenguatge d'etiquetatge escalable**.

XRI *m* Vegeu **identificador de recursos escalables**.

Bibliografia

"Authentication, Authorization and Accounting (aaa)" (s. d.). *IETF*. *Retrieved 2011-22-07*. <http://datatracker.ietf.org/wg/aaa/charter/>

Backhouse, J. (2006). "Interoperability of identity and identity management systems. Data Protection and Data Security". *Datenschutz und Datensicherheit* (vol. 30, núm. 9, pàg. 568-570).

Backhouse, J.; Halperin, R. (2008). "Approaching interoperability for identity management systems". A: K. Rannenberg; D. Royer; A. Deuker. *The Future of Identity in the Information Society: challenges and opportunities* (pàg. 245-268). Berlín/Heidelberg: Springer.

Baldwin, A.; Casassa, M.; Beres, Y.; Shiu, S. (2008). "On Identity Assurance in the Presence of Federated Identity Management Systems". A: *Actas del 2007 ACM Workshop on Digital Identity Management* (*DIM'07*) (pàg. 27-35). Alexandria, VA, EUA.

EduRoam (s/d). *EduRoam*. [Data de consulta: 22 de juliol de 2011]. <http://www.eduroam.es>

Harris, S. (2008). *All in one CISSP Exam Guide*. Nova York: McGraw-Hill.

Internet 2 (s/d). *Internet 2*. [Data de consulta: 7 de juliol de 2011]. <http://www.internet2.edu>

Internet 2 (s/d). *Shibboleth*. [Data de consulta: 7 de juliol de 2011]. <http://shibboleth.internet2.edu>

(ISC)2 (2007). *Official (ISC)2 Guide to the CISSP CBK*. Nova York: Auerbach Publications.

Keuleuven University (s/d). *SESAME in a Nutshell*. [Data de consulta: 19 de juny de 2011]. <http://www.cosic.esat.kuleuven.be/sesame/html/sesame_what.html>

Liberty Alliance (s/d). *Liberty Alliance Project*. [Data de consulta: 7 de juliol de 2011]. <http://projectliberty.org/liberty/content/download/3063/20504/file/liberty-cross-framework-v1.1.pdf>

Melgar, D.; Hondo, M.; Nadalin, A. (2002). *Web Services Security: Moving up the stack*. IBM DeveloperWorks. [Data de consulta: 22 de juliol de 2007]. <http://www.ibm.com/developerworks/library/ws-secroad/>

MIT (s/d). *MIT Kerberos Papers and Documentation web page*. [Data de consulta: 19 de juny de 2011]. <http://web.mit.edu/kerberos/papers.html>

OASIS (s/d). *OASIS*. Advanced Open Standards for Information Society. [Data de consulta: 22 de juliol de 2011]. <http://www.oasis-open.org>

Open PERMIS Project (s/d). *Open PERMIS Project*. [Data de consulta: 22 de juliol de 2011]. <http://www.openpermis.org/>

Proyecto HERAS (s/d). *Proyecto HERAS*. [Data de consulta: 22 de juliol de 2011]. <http://www.herasaf.org/>

RFC 2865 (s/d). *IETF*. [Data de consulta: 22 de juliol de 2011]. <http://www.faqs.org/rfcs/rfc2865.html>

Satchell, C. i altres (2006). *Knowing me Knowing You - User Perceptions of Federated Digital Identity Management Systems*. Suècia: ECIS.

Stoneburner, G. (s/d). *Underlining Technical Models for Information Technology Security. Recommendations of the National Institute of Standards and Techonology*. [Data de consulta: 19 de juny de 2011]. <http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf>

Windley, P. J. (2005). *Digital Identity*. Sebastopol, CA: O'Reilly Media, Inc.