**Universitatea „Politehnica” Timişoara**
**Facultatea de Automatică şi Calculatoare**

# Sisteme Single Sign-On sub atacuri Denial of Service

## Studiu de caz: Proiectul Liberty Alliance

**Referat nr. 3**

**Doctorand:**
ing. Valer BOCAN

**Conducător ştiinţific:**
prof. dr. ing. Vladimir CREŢU

**2005**

Ultima versiune a acestui document se poate obține de la adresa *http://www.dataman.ro*
sau scriind autorului la adresa de e-mail *vbocan@dataman.ro*

# Cuprins

# Capitolul 1

# Introducere

Cele mai multe companii au probleme legate de securitatea rețelei deoarece există o multitudine de aplicații care necesită autentificarea individuală a utilizatorilor. Situația se agravează pe măsură ce numărul aplicațiilor conectate creşte, iar nevoile specifice de autentificare evoluează. Nevoia de conectivitate a dus la o dependență crescută de asigurarea identității utilizatorului, situație care comportă o sumă de probleme unice din punctul de vedere al securității şi confidențialității.

## 1.1. Furtul de identitate

Furtul de identitate survine când un individ sustrage datele de identificare al unui alt individ în scopul folosirii acestora în mod fraudulos, în mod tipic pentru câştig economic [LIBE04a]. Răufăcătorii curajoşi fură articole ca portofele, genți, scrisori care deseori conțin şi informații financiare sau de credit. Alții caută printre deşeuri şi reuşesc să găsească articole cu conținut similar. Hoții moderni încearcă să fure date cu care să comită ulterior fraude prin Internet. Un furt de succes este însă o combinație a celor două metode, în sensul că hoțul îşi procură articole fizice pe diverse căi iar infracțiunea propriu-zisă este completată cu date disponibile pe Internet.

Înarmat cu astfel de informații, persoana poate deschide conturi în bancă, se poate abona la rețele de telefonie mobilă, reuşind să păcălească sistemele de securitate tradițională. Mai mult, facturile şi scrisorile nu ajung la victimă, acestea fiind „rerutate" la o adresă falsă. Situația poate să treneze un timp relativ lung, deoarece victima nu bănuieşte nimic, iar adevărata dimensiune a fraudei nu se arată decât atunci când întâmplător victima efectuează verificări asupra contului propriu.

## 1.2. Necesitatea sistemelor de tip Single Sign-On

Furtul de identitate în sine nu poate fi înlăturat. Încercări de fraudă au existat şi vor exista în continuare. Măsurile pe care societatea trebuie să le adopte vizează securizarea tranzacțiilor financiare on-line şi ameliorarea celei mai slabe verigi din lanțul creat de client şi furnizorul de servicii (SP). Această verigă este reprezentată de cele mai multe ori

de către autentificarea părților, uzual prin intermediul unei banale parole sau mai rar prin metode mai evoluate cum ar fi smart-card-ul sau autentificarea biometrică.

Odată cu proliferarea serviciilor on-line a apărut şi nevoia firească de autentificare a consumatorilor acestor servicii. Astfel, s-a ajuns ca un individ să fie nevoit a memora mai multe zeci de parole, lucru care tinde să devină ineficient şi mai cu seamă nesigur. Utilizatorii în general preferă să nu ia în calcul riscurile la care se expun şi pentru a evita notarea parolelor pe diverse suporturi aleg calea cea mai la îndemână, parola unică pentru toate serviciile.

O soluție la acest tip de probleme este adoptarea sistemelor cu autentificare singulară, numite Single Sign-On în literatura de specialitate. Această denumire o vom adopta şi noi pe tot parcursul lucrării de față. Pe scurt, un sistem SSO permite ca un utilizator să fie autentificat automat fiecărui serviciu pe care îl accesează, cu condiția autentificării inițiale cu succes a clientului la sistemul SSO.

## 1.3. Beneficii pentru companii

Companiile sunt cele mai afectate din punct de vedere economic din cauza furtului de identitate. În consecință, acestea sunt şi cele mai interesate să implementeze sisteme care să limiteze amploarea acestui fenomen. Cele mai multe companii (denumite uneori furnizori de servicii) autentifică clientul printr-un mecanism care implică un act emis de către un organism guvernamental (carte de identitate, paşaport, permis de conducere, etc.). Acest mecanism are neajunsul că este static (adică se revocă într-o manieră dificilă) şi mobil, adică în caz de furt acesta poate fi prezentat furnizorilor fără a trezi suspiciuni.

Sistemele SSO pe de altă parte beneficiază de versatilitatea extremă oferită de comunicațiile on-line, minimizând riscurile în caz de furt. În acest caz, identitatea consumatorului este protejată printr-un pseudonim care este valabil doar în dialogul dintre serviciul de autentificare şi furnizorul de servicii. Acest pseudonim nu poate fi folosit în dialogul cu alt furnizor de servicii, vorbim deci de o valabilitate limitată la un anumit domeniu. Mai mult decât atât, folosirea pseudonimelor permite un grad mare de anonimizare a serviciilor, adică un furnizor nu poate şti de ce alte servicii a mai beneficiat clientul său. Pseudonimele se pot schimba foarte uşor în caz de nevoie, consumatorul putând să nici nu fie conştient de această operație.

Folosirea sistemelor SSO mai prezintă un avantaj deloc de neglijat: dacă un consumator de servicii reclamă că anumite acțiuni au fost efectuate fraudulos la un furnizor în numele său, se poate verifica imediat dacă autentificarea s-a realizat în conjuncție cu un furnizor de identitate sau s-a realizat local, la furnizorul de servicii. Având în vedere că satisfacția clientului primează, investigația trebuie facilitată de organisme în drept, cu un minim impact asupra clientului în sine.

## 1.4. Beneficii pentru clienți

Deşi funcționarea unui sistem SSO presupune existența unor contracte şi relații de încredere între furnizorii de servicii şi cei de identitate, legătura între identitățile disparate nu se realizează decât cu acordul expres exprimat de către utilizator. Dacă în cazul companiilor pseudonimul oferă protecție în cazul unor litigii, în mod similar acesta reduce riscul de fraudă pe partea clientului. Compromiterea unei identități nu duce automat la compromiterea celorlalte.

În cadrul unui sistem SSO, relațiile de afaceri şi de încredere mijlocesc colaborarea între părți în cazul unui litigiu, iar clientul este cel care are întotdeauna de câştigat. În contrast, un sistem centralizat de autentificare, ca de exemplu un sistem guvernamental nu poate oferi garanția rezolvării litigiilor cu acelaşi succes deoarece

entitățile participante sunt investigate separat, existând riscul pierderii de informații sau indicii vitale pentru buna desfăşurare a anchetei.

## 1.5. Siguranța sistemelor SSO

Adoptarea pe scară largă a sistemelor SSO aduce cu sine întrebări referitoare la noul context în care se efectuează autentificarea utilizatorilor. O excelentă prezentare a argumentelor în favoarea SSO se face în [LIBE04a], idei pe care le preluăm în cele ce urmează:

Sisteme de tip Single Sign-On (SSO) au următoarele beneficii:

- Securitate şi confidențialitate sporite în tranzacțiile dintre client şi furnizorii de servicii şi cei de identitate
- Nu există un singur punct vulnerabil, adică în orice punct există o cantitate limitată de informații
- Acces la atributele clientului bazat pe listă de permisiuni.
- Actualizări ale documentației şi protocoalelor pentru a ține pasul cu atacurile şi vulnerabilitățile descoperite
- Răspuns imediat la incidente

Fiecare dintre aceste puncte este detaliat în cele ce urmează:

### Securitate şi confidențialitate sporite

Întreprinderile care adoptă specificațiile unui sistem SSO aderă la standarde cu un înalt grad de securitate. În acest fel, entitățile care comunică beneficiază de acelaşi nivel de securitate care reduce riscul breşelor de securitate prin captura de pachete şi alte atacuri comune.

### Lipsa unui singur punct vulnerabil

Din moment ce informațiile despre atribute şi identitate sunt distribuite în modelul de federare, nu există un loc unic unde sunt păstrate acestea. Drept urmare în cazul unui atac asupra bazei de date se compromite doar o parte a acestor informații, acțiunea neavând urmări catastrofice. Datele de autentificare pentru un site nu sunt neapărat folositoare şi altui site cum este cazul unui sistem centralizat deoarece nu se aplică principiul tranzitivității. Spre exemplu, dacă compania A are un contract de federare cu compania B, iar aceasta din urmă are la rândul ei un contract de federare cu compania C, acest lucru nu înseamnă că există implicit o federare a identității între A şi C.

### Acces bazat pe listă de permisiuni

Accesul la atributele utilizatorului se face pe bază de permisiuni. În acest fel expunerea atributelor este minimă iar fiecare furnizor de servicii are acces la un subset aprobat de utilizator.

### Actualizări ale documentației

Breşele de securitate sunt inevitabile. Utilizatorii care aderă la specificațiile unui sistem SSO sunt întotdeauna la curent cu ultimele specificații şi descoperiri în domeniu, fiind astfel protejați de riscuri. Toate breşele de securitate sunt probleme potențiale, dar experiența combinată a participanților la crearea sistemului SSO asigură un răspuns rapid şi eficace.

### Răspuns imediat la incidente

Cadrul de colaborare între furnizorii de identitate şi cei de servicii include protocoale de urmat în cazul producerii unui eveniment nedorit. Cadrul de colaborare este astfel o bază a încrederii şi cooperării.

# Capitolul 2

# Sisteme SSO

În prezent, utilizatorii care doresc să acceseze servicii on-line trebuie să folosească o sumă de credențiale (perechi nume utilizator şi parolă) pentru a accesa fiecare serviciu la care sunt înregistrați. Această stare de fapt are neajunsuri mari din punct de vedere al securității, deoarece utilizatorii se dovedesc incapabili a reține un număr mare de parole de calitate, în plus costurile apelurilor la help desk sunt importante tocmai din acest motiv.

Un sistem SSO abordează problema într-o manieră diferită. Utilizatorul se autentifică sistemului SSO cu credențialele proprii, apoi orice acces ulterior la servicii este automat autentificat de către sistemul SSO, fără intervenția manuală a utilizatorului. În prezent există mai multe arhitecturi SSO pe care le vom prezenta în continuare, fiecare cu avantajele şi dezavantajele sale [PASH03].

## 2.1. Tipuri de sisteme SSO

Sistemele SSO permit autentificarea automată a utilizatorilor către furnizorii de servicii. Autentificarea în sine implică identificare, deci un sistem SSO trebuie să aibă suport pentru managementul credențialelor utilizatorului. Întrucât acestea pot lua diverse forme, ne vom referi la acestea prin sintagma „identități SSO".

Putem deosebi două tipuri de sisteme SSO. Primul dintre acestea, numit „pseudo-SSO", funcționează ca un strat intermediar între utilizator şi aplicațiile pe care acesta le accesează. Autentificarea inițială (numită *autentificare primară*) se face între utilizator şi sistemul SSO. La fiecare acces la o aplicație care necesită credențialele utilizatorului, sistemul SSO intervine şi efectuează paşii necesari pentru ca autentificarea să fie reuşită. Pentru fiecare acces la aplicație se efectuează o autentificare într-o modalitate transparentă pentru utilizator. Aplicația poată să nu fie informată că are loc o autentificare comandată de către sistemul SSO.

Întrucât credențialele SSO sunt specifice fiecărui furnizor de servicii (aplicații software, etc.) relația identitate SSO/furnizor este *n : 1*, adică orice identitate dată corespunde unui singur furnizor.

Schema de principiu al unui sistem „pseudo-SSO" este prezentată în figura 1.

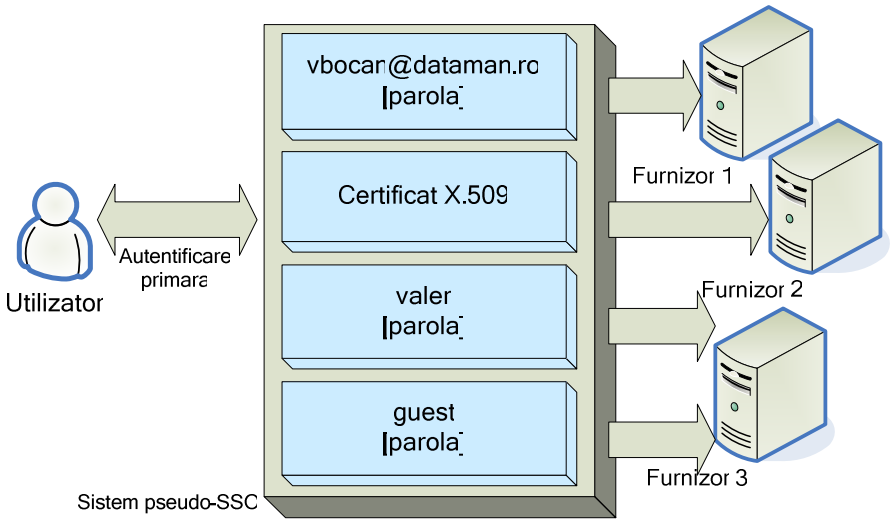


Figura 1. Mecanismul de principiu al unui sistem „pseudo-SSO”

La modul concret, un sistem „pseudo-SSO” este un program care monitorizează comportamentul unor produse software care în mod tradițional ar cere autentificarea utilizatorului în diverse moduri, cum ar fi ferestre de login, prompt, argumente în linia de comandă, etc. Prin mecanisme specifice, aceste cereri de autentificare sunt interceptate şi deservite în funcție de credențialele înregistrate pentru produsul în cauză, pentru utilizatorul autentificat curent. Din punct de vedere al mecanismelor de implementare, sistemele „pseudo-SSO” nu sunt altceva decât mecanisme de automatizare ale unor operații care în mod obişnuit ar fi efectuate de către utilizator. La fiecare acces la aplicație se face o autentificare în sine.

Dezavantajul major al acestui tip de sisteme SSO este că credențialele se mențin în clar, deoarece acestea sunt necesare a fi livrate tot în clar furnizorilor de servicii. Protejarea credențialelor în drumul lor de la sistemul SSO la aplicație este foarte dificil capturarea putându-se face prin diverse mijloace nu foarte avansate. Un alt dezavantaj este necesitatea actualizării scripturilor sau modulelor de interceptare a ferestrelor de autentificare pentru fiecare produs monitorizat în parte. De regulă există un interval „mort” în care sistemul „pseudo-SSO” nu are cunoştințe despre noile versiuni ale produselor, timp în care clienții trebuie să aştepte. În mod clar, această stare de fapt este un neajuns, iar pentru a-şi păstra clienții, companiile care dezvoltă sisteme „pseudo-SSO” trebuie să fie în permanentă alertă, ceea ce duce la un consum crescut de resurse şi implicit la un cost ridicat.

Avantajul acestui tip de sisteme SSO este că pentru implementarea sa sunt necesare schimbări minime în infrastructura software deoarece aplicațiile monitorizate nu trebuie să fie informate de noul sistem în funcțiune.

Cel de-al doilea tip de sisteme SSO este fundamental diferit de cel anterior şi îl vom numi în continuare „true-SSO”. În acest caz, autentificarea primară se realizează către o componentă numită *Authentication Service Provider (ASP)*. Uneori această componentă este numită *Identity Service Provider (ISP)*. Pentru a realiza SSO, ASP trebuie să aibă stabilită o relație cu fiecare furnizor de servicii (SP), relație care de regulă este consfințită printr-un contract. De asemenea trebuie să existe un canal securizat de comunicație între ASP şi SP.

Odată ce autentificarea primară a avut loc, la cererea furnizorilor de servicii (SP), ASP-ul realizează autentificarea prin intermediul unui protocol dedicat, folosind aşa-

numitele *authentication assertions*. Acestea conţin identitatea utilizatorului şi starea sa aşa cum este cunoscută de către ASP şi trebuie transmise în mod securizat, în funcţie de specificul implementării. Spre deosebire de sistemele pseudo-SSO, relaţia identitate SSO / furnizor este *n : m*. Aceasta înseamnă că utilizatorul poate alege dintr-o plajă de identităţi pentru orice SP ales, dar aceeaşi identitate poate fi utilizată cu mai mulţi SP, dacă utilizatorul dispune acest lucru. Astfel este posibilă atribuirea de roluri specifice identităţilor SSO (care sunt de fapt pseudonime, aşa cum au fost definite în [PFIT01]).

Schema de principiu a unui sistem true-SSO este prezentată în figura 2.

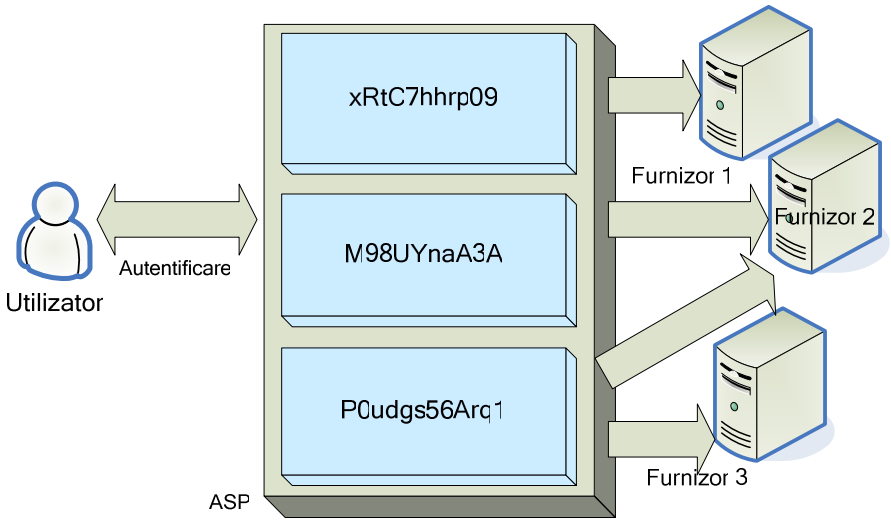


Figura 2. Mecanismul de principiu al unui sistem „true-SSO”

La modul concret, ASP este un serviciu care deserveşte cereri provenite de la clienţi care au cunoştinţă de faptul că un sistem SSO este în funcţiune. Dezavantajul major al acestui sistem este necesitatea modificării produselor software. Acestea trebuie (re)proiectate şi implementate urmând specificaţiile sistemului SSO în cauză, lucru care nu este tocmai uşor pentru aplicaţii *legacy*.

Avantajul de necontestat este că autentificarea se face o singură dată, credenţialele nu sunt stocate şi transmise în clar, putându-se totodată implementa anonimizarea, adică imposibilitatea creării de conexiuni între identităţile separate ale aceluiaşi utilizator fără ştirea acestuia.

Traseul informaţiei între părţile implicate este ilustrat în figura 3. În prima fază, utilizatorul efectuează autentificarea primară către ASP (1). Acest pas se efectuează o singură dată într-o sesiune de lucru. La un moment ulterior, utilizatorul doreşte să acceseze un serviciu oferit de un furnizor oarecare (2). Furnizorul trebuie să verifice la rândul său identitatea clientului, iar pentru acest lucru va solicita informaţiile de la ASP (3.1), care îi trimite un identificator ce reprezintă identitatea utilizatorului (3.2). În acest moment furnizorul de servicii poate trece la îndeplinirea iniţială a cererii utilizatorului (4).

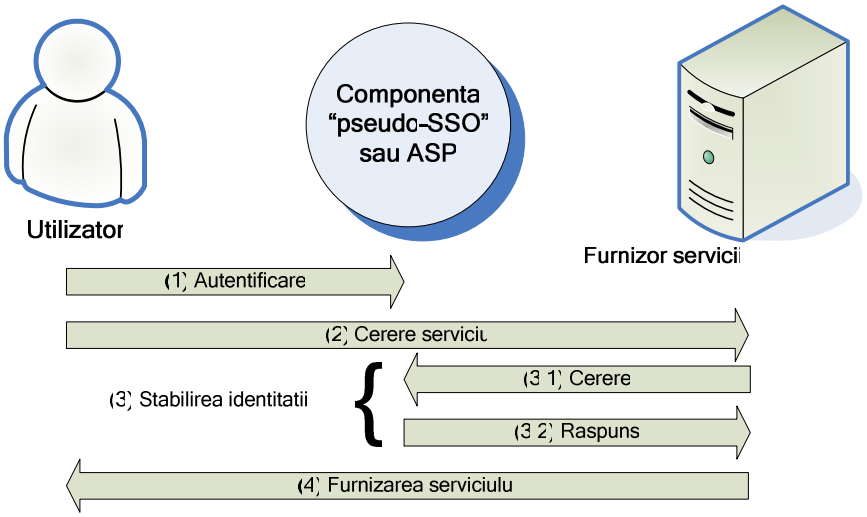


Figura 3. Fluxul informațional într-un sistem SSO generic

Modulele „pseudo-SSO"/ASP se pot afla pe calculatorul client (cel al utilizatorului) sau se poate afla pe o altă maşină pe care o numim „SSO proxy". Astfel, deosebim următoarele categorii de sisteme SSO [PASH03]:

- Sisteme pseudo-SSO locale
- Sisteme pseudo-SSO bazate pe proxy
- Sisteme SSO locale
- Sisteme SSO bazate pe proxy

În cele ce urmează vom revedea fiecare dintre aceste categorii în detaliu.

## 2.2. Sisteme pseudo-SSO locale

Într-un sistem pseudo-SSO local, componenta pseudo-SSO se află pe sistemul clientului. La începutul sesiunii de lucru, utilizatorul se autentifică componentei care tipic constă dintr-o bază de date care conține identitățile utilizatorului pentru fiecare serviciu înregistrat în parte. O proprietate caracteristică este necesitatea de a păstra credențialele în clar, de unde rezultă necesitatea ca utilizatorul să aibă încredere în sistemul pe care lucrează. De remarcat că această arhitectură nu se pretează la sisteme publice sau unde accesul nu este restricționat.

## 2.3. Sisteme pseudo-SSO bazate pe proxy

Într-un sistem pseudo-SSO local bazat pe proxy, componenta pseudo-SSO se află pe un sistem separat de cel al clientului. Autentificarea primară se efectuează la fel ca în cazul anterior la începutul sesiunii de lucru (eventual şi ulterior dacă proxy-ul doreşte reautentificarea periodică a clientului). Identitățile pentru fiecare furnizor de servicii în parte sunt stocate pe proxy, într-o bază de date securizată, însă tot este necesară prezentarea în clar a acestora în cazul dialogului cu un furnizor de servicii. Cererile de autentificare pot fi redirectate de către client către proxy printr-un mecanism special destinat acestui scop, fie sunt pur şi simplu interceptate fără ca programul în cauză să fie anunțat în vreun fel, astfel încât procesul să fie transparent pentru client. De remarcat că arhitectura se pretează la sistemele publice cu condiția ca proxy-ul să fie securizat corespunzător.

## 2.4. Sisteme SSO locale

Un sistem SSO local are ca trăsătură principală furnizorul de identitate (ASP – authentication provider) plasat pe sistemul client. Acesta, cunoscând identitatea clientului dintr-o autentificare primară anterioară, va transmite furnizorilor de servicii identitatea clientului printr-un canal securizat. Între ASP şi furnizorii de servicii trebuie să existe o relaţie de încredere stabilită printr-un contract.

Diferenţa esenţială faţă de sistemele psuedo-SSO este că ASP-ul realizează autentificarea printr-un protocol special, iar credenţialele nu sunt transmise niciodată în clar.

## 2.5. Sisteme SSO bazate pe proxy

În acest model, rolul ASP-ului este luat de un server extern clientului, care acţionează ca un proxy între client şi furnizorii de servicii. Avantajul faţă de modelul anterior este separarea fizică între client şi ASP, în aşa fel încât acesta din urmă poate fi mai bine protejat împotriva încercărilor de fraudare. Este important de arătat că un ASP poate impersona oricare identitate pe care o reprezintă, rezultă deci că încrederea clientului şi a furnizorilor de servicii este esenţială pentru buna funcţionare a sistemului.

## 2.6. Proprietăţi ale sistemelor SSO

Sistemele SSO au o serie de proprietăţi pe care le vom trece în revistă în această secţiune.

### Confidenţialitatea

Confidenţialitatea are o importanţă deosebită în sistemele SSO [WWWC02]. Procesul de autentificare nu trebuie să dezvăluie identitatea directă a clientului. De asemenea, este de dorit ca două sau mai multe identităţi distincte ale aceluiaşi client să nu poată fi corelate fără consimţământul acestuia. Imposibilitatea corelării identităţilor implică imposibilitatea determinării identităţii personale a clientului pe baza credenţialelor sale [PFIT01].

Sistemele pseudo-SSO nu pot garanta pseudonimitatea (necorelarea identităţilor), deoarece acestea sunt specifice fiecărui furnizor de servicii şi pot conţine uneori date personale ale clientului. Astfel, unele sisteme pot cere autentificarea cu certificate, site-urile FTP cer uneori ca numele utilizatorului să fie chiar adresa e-mail, sau – cel mai grav – codul numeric personal.

### Acces anonim la reţea

Deşi confidenţialitatea este o cerinţă a oricărei implementări SSO, aceasta depinde de modul particular în care este exploatat sistemul. Comunicarea între părţile implicate se realizează pe căi de comunicaţie „punct la punct”, deci fiecare nod trebuie identificat în mod unic printr-o adresă. Din nefericire, comunicaţia pe straturile inferioare ale protocoalelor se face fără anonimizare, iar un eventual atacator poate compromite anonimitatea prin analiza pachetelor [BACK01]. Fiecare pachet are în componenţă printre altele adresa sursă care poate oferi informaţii despre poziţionarea geografică a furnizorului de Internet. Cu această informaţie, atacatorul poate realiza corelarea identităţilor fără un efort semnificativ.

Problema determinării sursei traficului este rezolvată prin folosirea unor proxy-uri de anonimizare (ascundere a identităţii) prin care se realizează comunicaţia între client şi furnizorul de servicii. Proxy-ul are rolul de a înlocui adresa clientului cu propria sa adresă (posibil şi alte informaţii personale), făcând astfel imposibilă identificarea clientului.

Nivelul de anonimitate este dat în cele din urmă de numărul de utilizatori care folosesc proxy-ul [PFIT01]. Exemple de astfel de servicii sunt Anonymizer [WWW1] şi Freedom WebSecure [WWW2].

Modelul de anonimizare cu un singur proxy nu protejează împotriva analizei de trafic şi se poate folosi atât timp cât operatorul proxy-ului nu este compromis. Din fericire există scheme – ca cele descrise în [CHAU81, GOLD99] – care protejează împotriva analizei traficului şi distribuie încrederea între proxy-urile implicate. O implementare de acest fel este JAP [WWW3].

Anonimizarea poate fi realizată şi la nivelul proxy-ului SSO, într-o manieră transparentă pentru client şi pentru furnizorul de servicii. Desigur, întreg traficul clientului trebuie în acest caz rutat prin proxy, însă clientul nu va trebui să aibă încredere într-o altă entitate decât cea în care are deja încredere.

### Mobilitatea utilizatorului

Sistemele SSO bazate pe proxy, prin însăşi structura lor permit ca utilizatorul să fie mobil, deoarece baza de date cu credenţialele sale sunt pe o maşină alta decât cea a clientului. Autentificarea se poate realiza de oriunde, în limitele impuse de protocolul de autentificare în sine.

Sistemele SSO locale oferă mobilitate utilizatorului în măsura în care baza de date cu credenţialele utilizatorului sunt păstrate extern clientului. Gradul de siguranţă este determinat în principal de modul de stocare al acestora, criptat sau în clar. Cele mai cunoscute implementări comerciale sunt Novell Secure Login (www.novell.com/products/securelogin), Passlogic V-Go (www.passlogic.com/sso), Protcom SecureLogin (www.protcom.cc), RSA ClearTrust (www.rsasecurity.com).

### Utilizarea în medii ostile

În unele cazuri este necesar ca furnizorii de servicii să fie accesaţi din medii ostile, cum ar fi Internet café-urile şi terminalele publice. Este de dorit ca maşina clientului să nu aibă acces la secretele memorate în sistemul SSO, pentru a evita lansarea de atacuri prin impersonare (asumarea unei identităţi false). În acest caz este utilă implementarea modelelor cu proxy. Autentificarea primară trebuie în acest caz să se realizeze prin protocoale de autentificare adecvate, rezistente la atacuri replay sau man-in-the-middle.

### Costuri de instalare şi întreţinere

La capitolul costuri, instalarea şi întreţinerea sistemelor pseudo-SSO şi true-SSO sunt diferite. În cazul sistemelor pseudo-SSO, costurile de instalare sunt mici deoarece nu trebuie să existe o infrastructură de securitate propriu-zisă, iar furnizorii de servicii nu trebuie să aibă cunoştinţă de noul modul. Pe de altă parte, dacă furnizorul de servicii schimbă modul de autentificare al clientului, acest lucru trebuie să se reflecte în componenta pseudo-SSO, ceea ce duce la cheltuieli de exploatare. Lucrurile sunt cu atât mai complicate cu cât mediul în care sistemul pseudo-SSO este instalat este mai dinamic.

Situaţia este exact inversă în cazul sistemelor true-SSO. Cheltuielile legate de instalare sunt mari întrucât este necesară implementarea unei infrastructuri pentru ca dialogul client – furnizor de servicii să se realizeze în mod securizat. Acolo unde se impune este de asemenea necesară semnarea de contracte de servicii şi tratate de încredere reciprocă a părţilor implicate. Costurile de operare sunt în schimb reduse, deoarece schimbările părţilor nu implică automat modificarea interfeţei SSO.

### Costuri de exploatare

Costurile de exploatare sunt mai mici în cazul sistemelor SSO locale față de cele bazate pe proxy, deoarece se induc costuri legate de operarea proxy-ului. Comunicația între părțile implicate poate fi taxată după un model negociat, mai ales când tot traficul clientului este rutat fizic prin proxy.

### Relații de încredere

Atât în cazul pseudo-SSO cât şi în cazul true-SSO, clientul trebuie să aibă încredere în componenta care stochează credențialele. În cazul sistemelor bazate pe proxy, relația de încredere se poate reglementa prin contracte și politici la diferite niveluri pentru ca integritatea sistemului SSO să fie menținută. În cazul sistemelor SSO locale, utilizatorul trebuie să aibă încredere în componenta software care stochează şi prelucrează credențialele.

Dacă la sistemele true-SSO am putut evidenția în mod clar posibilitatea stabilirii prin contract a relației de încredere, în cazul pseudo-SSO relația de încredere este difuză, deoarece furnizorul de servicii poate să nu aibă cunoştință despre chiar existența sistemului SSO. Baza de date cu credențiale poate fi stocată criptat sau în clar sau poate fi replicată pe mai multe servere, de unde rezultă că relația de încredere între client, furnizorul de servicii şi cel de identitate depinde de implementarea concretă a sistemului SSO. Mai mult, această relație este dinamică, în sensul că se poate schimba odată cu modificarea metodei de autentificare a furnizorului de servicii.

### Rezolvarea conflictelor

În caz de conflict sau de cercetare legală, este necesar ca operatorul proxy-ului să poată furniza detalii despre tranzacțiile efectuate. Observația este valabilă atât în cazul sistemelor pseudo-SSO cât şi true-SSO, însă accentul cade pe sisteme true-SSO unde jurnalul trebuie să fie menținut la zi cu toate tranzacțiile efectuate deoarece sunt implicate semnificativ mai multe entități decât în cazul pseudo-SSO.

### Medii închise şi medii deschise

Mediile închise în general nu au o politică de confidențialitate bine conturată sau dacă există aceasta este considerată de importanță redusă. În acest caz accentul se pune pe costurile de achiziționare, exploatare şi întreținere a sistemului SSO. Deoarece aceste costuri sunt mai mici în cazul sistemelor pseudo-SSO, sistemele scalabile la nivel de întreprindere sunt cele mai potrivite pentru recuperarea rapidă a investiției. Arhitectura acestor sisteme este analizată în [CLER02].

În mediile deschise, nevoia de confidențialitate este acută. Nivelul de confidențialitate cerut este oferit doar de către sistemele true-SSO, astfel încât costurile legate de operare şi întreținere sunt depăşite.

# Capitolul 3

# Federarea identității

În zilele noastre, Internetul este vehiculul primar pentru afaceri, comunitate şi interacțiune personală. Noțiunea de *identitate* este componenta crucială a acestui vehicul. Astăzi, identitatea unui individ dat este fragmentată în mai multe experiențe izolate de tipul client – afacere [LIBE03a].

În acest context vorbim despre *federarea identității*, adică crearea unei legături logice între insulele de identitate aflate la diverşi furnizori de servicii pentru îmbunătățirea experienței utilizatorului în ceea ce priveşte comoditatea autentificării.

## 3.1. Identitatea în rețea

Când utilizatorii interacționează cu furnizorii de servicii, aceştia îşi personalizează modul de lucru. Spre exemplu, o persoană îşi alege numele de utilizator şi parola pentru un anume serviciu, apoi optează pentru atributele pe care sistemul le afişează şi forma în care le afişează.

După cum arată figura 4, identitatea în rețea pentru un utilizator anume este dată de suma atributelor sale distribuite la furnizorii de servicii pe care îi agreează. Fiecare furnizor este în posesia atributelor care să îi permită livrarea serviciului în condiții bune, iar în unele cazuri atributele cerute sunt chiar mai multe decât ar fi în mod normal necesar. Se ajunge astfel în situația ca doi furnizori diferiți să dețină aceleaşi atribute ale utilizatorului.

Misiunea unui sistem de federare a identității cuprinde [LIBE03a]:

- Permite utilizatorilor să-şi protejeze identitatea în rețea
- Permite companiilor să mențină contactul cu clienții fără intervenția unui terț (furnizorul de identitate nu mai intervine în schimbul de mesaje odată ce autentificarea primară a fost efectuată cu succes)
- Furnizarea unui sistem single sign-on care include autentificare şi autorizare descentralizate, de la mai mulți furnizori de identitate
- Crearea unei infrastructuri care să permită interconectarea dispozitivelor de rețea actuale şi viitoare

Cerințele enunțate anterior se pot îndeplini atunci când companiile se organizează în *cercuri de încredere* şi pe acorduri operaționale care definesc *relațiile de încredere*

între companii. Mai mult, utilizatorii înşişi trebuie să fie dispuşi să federeze conturile separate de la fiecare furnizor de servicii (identitatea locală). Pe scurt, un cerc de încredere reprezintă o federare de furnizori de servicii şi identitate care au relaţii de afaceri şi acorduri operaţionale, cu care clienţii pot face afaceri într-un mediu securizat şi aproape transparent.

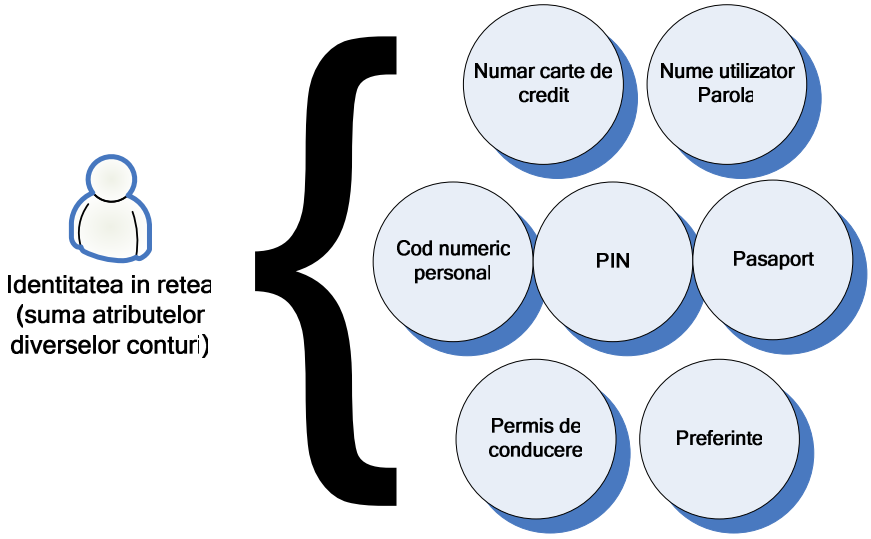


Figura 4. Identitatea în reţea este constituită din suma atributelor utilizatorului

## 3.2. Federarea identităţii şi autentificarea unică (SSO)

Furnizorii de servicii sunt practic orice entităţi cu prezenţă pe Internet care oferă un serviciu oarecare. Exemple de astfel de furnizori sunt: portaluri pe diverse teme, comercianţi, transportatori, instituţii financiare, instituţii guvernamentale, instituţii medicale, organizaţii non-profit, ş.a.

Furnizorii de identitate creează premisele cercului de încredere prin oferirea de servicii de identificare. Furnizorii de servicii şi cei de identitate stabilesc împreună un cerc al încrederii (ca în figura 5) în care atributele utilizatorilor circulă sub controlul direct al acestora. Spre exemplu, un furnizor de identitate poate oferi servicii de autentificare pentru angajaţii unei companii, sau o bancă poate permite anumitor companii accesul la atributele clienţilor săi, în perspectiva oferirii de servicii cu valoare adăugată. Uneori o organizaţie poate fi atât furnizor de servicii cât şi furnizor de identitate.

Pentru ca adoptarea sistemului SSO să se facă uşor, este de dorit ca utilizatorul să interacţioneze cu furnizorul de servicii printr-un simplu browser Internet (cu sau fără suportul de JavaScript instalat). Un client care are la dispoziţie doar un browser se numeşte „client uşor” sau „zero-footprint”.

Cel mai reprezentativ sistem SSO este reprezentat de proiectul Liberty Alliance, la care vom face referire în secţiunile care urmează. Secţiunea din Liberty Alliance care se ocupă cu federarea identităţii se numeşte ID-FF (Identity Federation Framework) [LIBE03a].

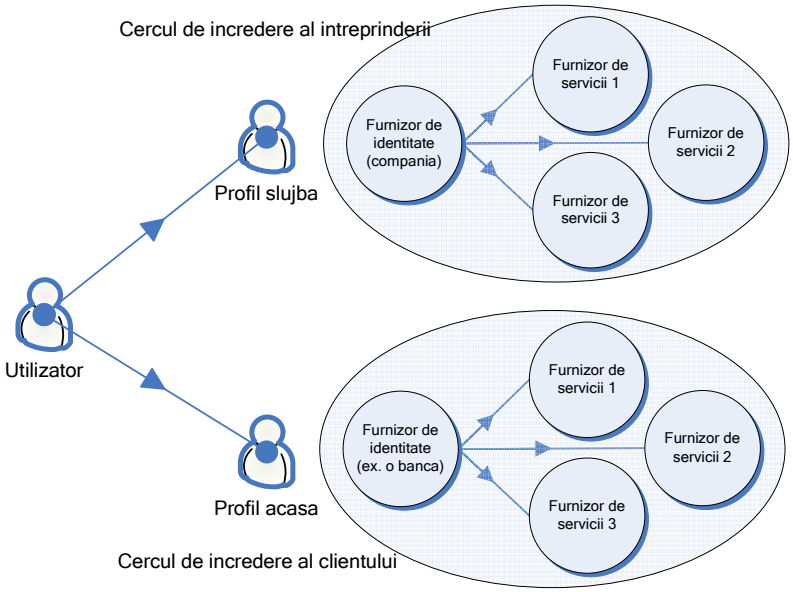


Figura 5. Identitatea federată în rețea şi cercuri de încredere

## 3.3. Exemple

În această secțiune vom prezenta două exemple de utilizare din perspectiva utilizatorului, preluate după [LIBE03a].

Exemplele se bazează pe următorul set de actori:

- **Joe Self** – Un utilizator de servicii web
- **Airline.inc** – O companie aeriană ce menține relații cu un grup de parteneri. Compania este furnizor de identitate.
- **CarRental.inc** – O companie de închirieri de maşini ce are relații cu compania aeriană. Compania este furnizor de servicii.

ID-FF prezintă două componente: federarea identității şi single sign-on.

### Federarea identității

Procesul începe cu navigarea lui Joe Self la site-ul web al Airline.inc, unde acesta îşi introduce numele şi parola (figura 6). Utilizatorul se spune că este autentificat în momentul în care validarea credențialelor acestuia s-a efectuat cu succes.

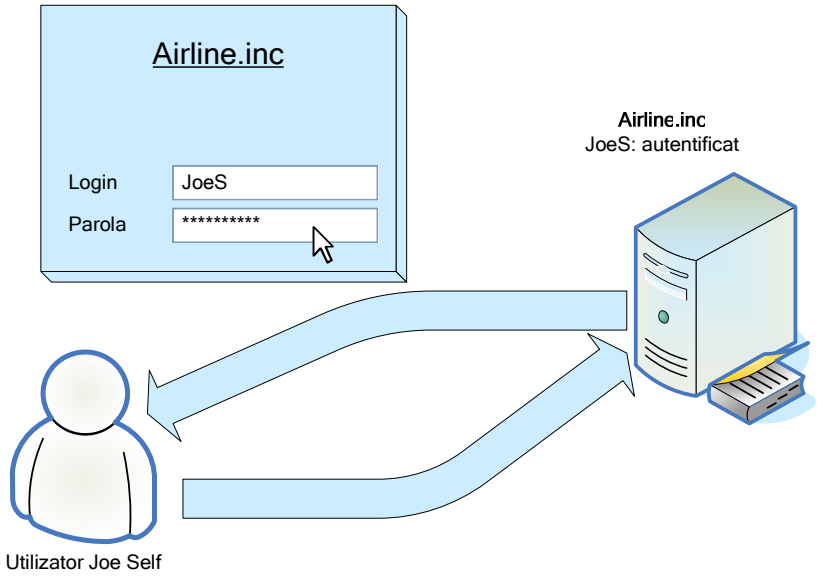


Figura 6. Un utilizator se autentifică unui site care foloseşte protocolul Liberty Alliance

Airline.inc (ca de altfel oricare alt furnizor de identitate) anunţă utilizatorii eligibili asupra posibilităţii de federare a identităţilor locale membrilor cercului de încredere. Federarea se va efectua doar cu consimţământul expres al utilizatorului.

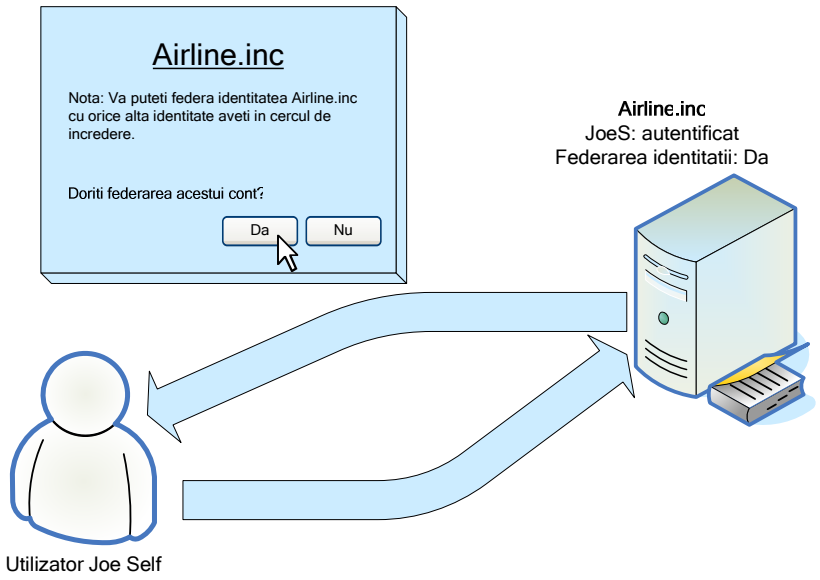


Figura 7. Unui utilizator autentificat i se cere acceptul pentru federarea identităţilor sale din cercul de încredere al furnizorului de identitate

Este important de remarcat faptul că procesul descris în figura 7 reprezintă doar acordarea consimţământului utilizatorului ca identitatea sa să fie prezentată celorlalţi membri ai cercului de încredere. Acest consimţământ nu înseamnă automat şi acceptul ca identitatea să fie federată.

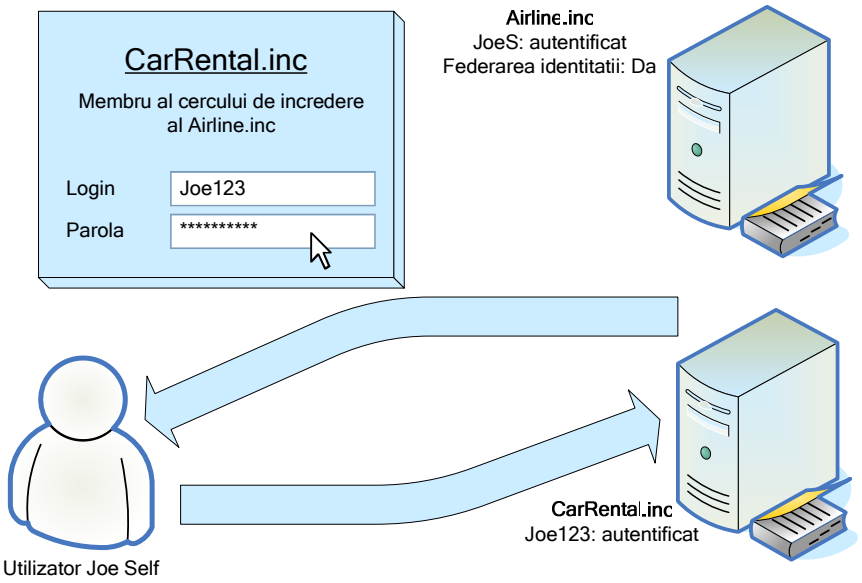


Figura 8. Utilizatorul se autentifică cu identitatea locală a unui furnizor de servicii

La un moment ulterior Joe Self poate vizita website-ul unui membru al cercului de încredere, spre exemplu CarRentals.inc, posibil urmând o legătură spre acesta. Dacă furnizorul de servicii menține un set de credențiale locale, acesta va cere autentificarea locală a utilizatorului (figura 8). Utilizatorului i se oferă posibilitatea de a crea o legătură între identitățile locale aflate la cei doi furnizori (Airline.inc şi CarRental.inc), numită federare (figura 9). Oferirea posibilității de federare se poate face înainte sau după autentificarea locală, în funcție de politica aplicată şi de implementarea de fapt.

După efectuarea cu succes a federării, Joe Self consumă serviciile oferite de CarRentals.inc în mod normal, ca înainte de federare. Acesta însă poate beneficia de facilități suplimentare, ca urmare a federării.

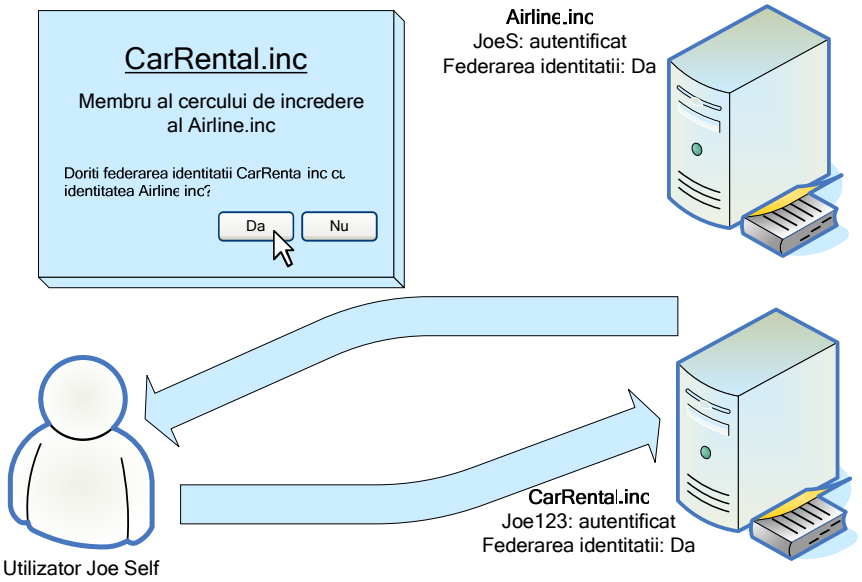


Figura 9. Utilizatorul este întrebat dacă doreşte federarea identităților

În figurile de până acum am folosit ca nume de utilizator identificatori în clar, ca JoeS şi Joe123. În realitate, pentru a evita problemele legate de confidențialitate şi relaționarea neautorizată a conturilor se vor folosi pseudonime.

### Single Sign-On

Single Sign-On este procesul prin care autentificarea primară efectuată la unul dintre membrii cercului de încredere este „replicată" automat fiecărui furnizor de servicii. Acest proces se bazează pe consimțământul expres al utilizatorului, dat în pasul anterior, federarea identității.

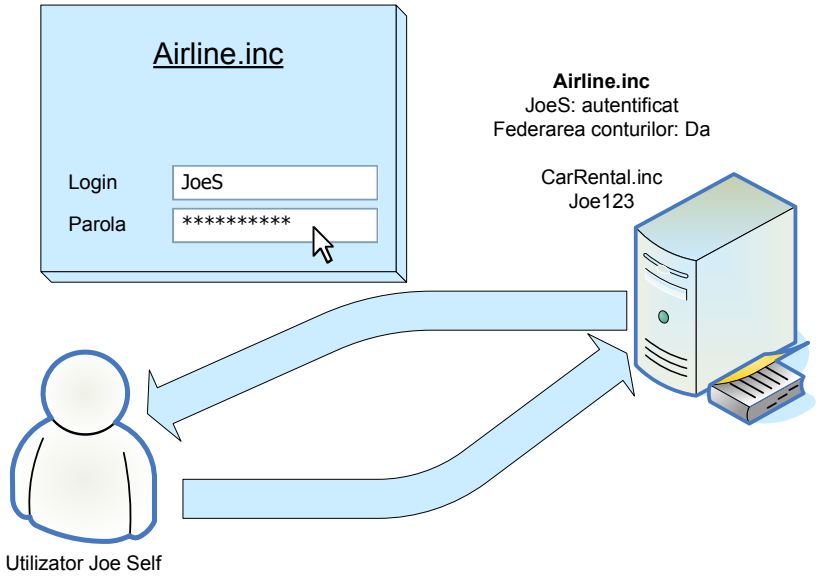


Figura 10. Un utilizator se autentifică unui site cu credențialele locale

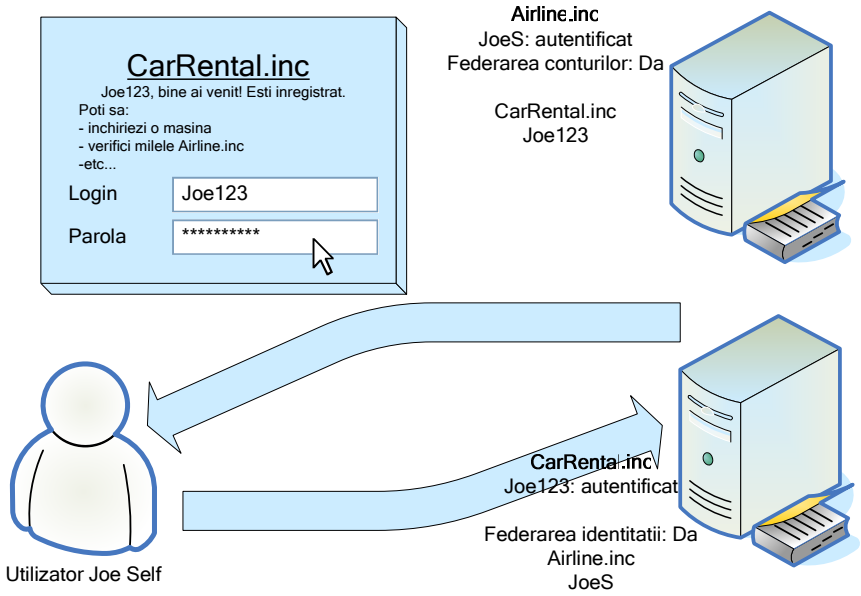


Figura 11. Utilizatorul autentificat pe site-ul Airline.inc poate beneficia automat de serviciile CarRentals.inc

Odată autentificat la site-ul Airline.inc, Joe Self va putea să navigheze către CarRental.inc şi să beneficieze de serviciile sale fără a fi nevoit să se reautentifice. Figurile 10 şi 11 ilustrează acest concept.

# Capitolul 4

# Protocoale Liberty

Suita de protocoale Liberty constă din următoarele:

- **Single Sign-On and Federation** – Protocolul prin care se federează identitatea utilizatorului şi prin care se obține autentificarea unică (SSO)
- **Name Registration** – Protocolul prin care un furnizor poate să înregistreze un identificator opac pentru un utilizator
- **Federation Termination Notification** – Protocolul prin care un furnizor poate notifica alt furnizor asupra terminării unei federări (acțiune cunoscută ca defederare)
- **Single Logout** – Protocolul prin care furnizorii se anunță reciproc asupra evenimentelor de logout.
- **Name Identifier Mapping** – Protocolul prin care furnizorii de servicii pot obține identificatori de nume corespunzători unei federări de identitate în care nu participă.

Acest capitol va detalia fiecare dintre aceste protocoale, aşa cum sunt ele descrise în [LIBE03b].

## 4.1. Cerințe generale

În această secțiune sunt definite un set de cerințe aplicabile tuturor protocoalelor.

### Semnătura XML

Specificația semnăturii XML defineşte un cadru general pentru semnarea documentelor XML. Toate documentele XML trebuie să adere la constrângerile din „XML Signature Provider" definite în [MALE03], din care am selectat câteva:

### Versionarea protocoalelor şi a declarațiilor

Informațiile de versiune apar în mesajele protocoalelor şi a declarațiilor. La momentul de față, orice element de protocol sau declarație care conține atributele `MajorVersion` şi `MinorVersion` trebuie să conțină valorile 1, respectiv 2.

În alte cazuri, specificația urmează cerințele de numerotare a versiunilor, regulile de procesare şi condițiile de eroare specificate în „SAML Versioning" din [MALE03].

Aceasta înseamnă că orice elemente definite de SAML care conţin atributele `MajorVersion` şi `MinorVersion` trebuie să conţină valorile 1, respectiv 1.

### Unicitatea identificatorului furnizorului şi afilierii

Toţi furnizorii şi afilierile au un identificator bazat pe URI (Uniform Resource Identifier). Acest identificator trebuie să fie unic în cadrul spaţiului format din toţi furnizorii cu care comunică. Se recomandă ca furnizorul să folosească URI-ul cu domeniul propriu în cadrul identificatorului. Identificatorul nu trebuie să depăşească 1024 de caractere.

### Construcţia identificatorului

Clienţilor li se atribuie identificatori de către furnizorii de servicii şi uneori de cei de identitate. Când este generat de către un furnizor de identitate, identificatorul trebuie construit cu valori aleatoare care nu au nici o legătură cu identitatea clientului. Intenţia este de a crea un pseudonim non-public care să împiedice descoperirea identităţii şi activităţii clientului. Furnizorii de servicii trebuie să urmeze aceleaşi reguli. Identificatorii necriptaţi nu trebuie să depăşească 256 de caractere.

La generarea identificatorilor pentru clienţi, în caz că se foloseşte o tehnică criptografică, se recomandă ca probabilitatea de apariţie a doi identificatori identici să fie mai mică decât $2^{-160}$.

### Verificarea semnăturii

Regulile de prelucrare a mesajelor pentru protocoalele ce vor fi descrise includ verificarea digitală a semnăturilor. În aceste cazuri nu este suficientă doar verificarea în sine a semnăturii, ci trebuie urmate regulile definite în [MALE03]. Spre exemplu, când se utilizează certificate X.509 v3 se recomandă validarea căii de certificare în acord cu profilul PKIX din [RFC3280].

### Securitatea

Aceste specificaţii exprimă doar idei generale despre fiecare protocol în parte, lăsând detaliile pe seama fiecărei implementări în parte. Consideraţii de securitate se găsesc la [CANT03].

### Valori de timp

Toate valorile de timp se bazează pe specificaţia din [BIRO02]. Acestea se exprimă în forma UTC, indicată de un Z ce urmează imediat valorii de timp.

Implementările nu trebuie să se bazeze pe rezoluţii mai mari de o secundă întrucât există numeroase aplicaţii care ignoră partea fracţionară a secundei.

Furnizorii nu trebuie să se bazeze pe sincronizarea temporală la mai puţin de un minut a două entităţi. Spre exemplu, un furnizor de identitate care comunică cu un furnizor de servicii nu trebuie să considere că timpul său local este la o diferenţă mai mică de un minut de timpul interlocutorului său.

## 4.2. Protocolul Single Sign-On and Federation

Protocolul de federare şi autentificare unică este un protocol de tip cerere-răspuns prin care au loc acţiunile de federare şi autentificare unică. Protocolul se desfăşoară între un furnizor de servicii şi unul sau mai mulţi furnizori de identitate. Protocolul este ilustrat în figura 12.

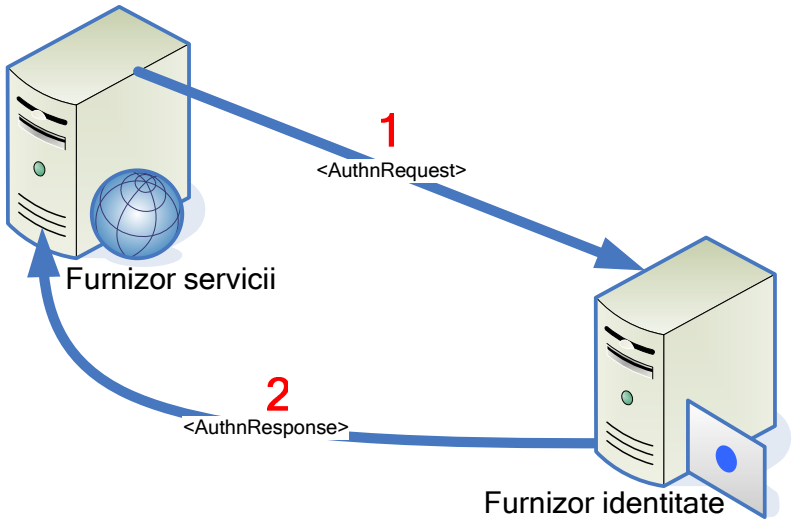


Figura 12. Schimbul de mesaje în protocolul de federare şi autentificare unică

1. Furnizorul de servicii trimite un mesaj `<AuthnRequest>` unui furnizor de identitate, căruia îi cere o declarație de autentificare. Opțional, furnizorul de servicii poate cere ca identitatea să fie federată.
2. Furnizorul de identitate răspunde cu declarația de identitate sub forma unui mesaj `<AuthnResponse>`. Opțional, furnizorul de identitate federează identitatea clientului aflată la furnizorul de servicii.

În caz că furnizorul de identitate nu poate executa singur operația, acesta poate trimite mai departe cererea către un alt furnizor de identitate pentru autentificare, caz în care protocolul se repetă între destinatarul mesajului `<AuthnRequest>` original şi alți furnizori de identitate.

### Cerere

Furnizorul de servicii generează o cerere `<AuthnRequest>` inițială. Această conține o serie de atribute prin care se poate controla comportamentul furnizorului de identitate. Autorul cererii poate controla următoarele comportamente:

- Prezentarea către client a unui ecran de autentificare, dacă acesta nu este autentificat deja
- Prezentarea către client a unui ecran de autentificare, chiar dacă acesta este autentificat deja
- Federarea identității de la furnizorul de identitate cu cea de la furnizorul de servicii
- Emiterea unui identificator anonim şi temporar în numele clientului, în dialogul cu furnizorii de servicii
- Folosirea unui profil specific al protocolului în răspunsul la cereri
- Folosirea unui context specific de autentificare (ex.: smart-card în loc de parolă)
- Restricționarea capacității ca destinatarul cererii să trimită mai departe cererea unor alți furnizori de identitate

În plus, furnizorul de servicii poate include în cerere orice informații de stare pe care furnizorul de identitate le trimite înapoi în răspuns (suport pentru protocoale „stateless").

Mesajul `<AuthnRequest>` trebuie semnat.

### Răspuns

Răspunsul furnizorului de identitate constă fie dintr-un element `<AuthnResponse>` conținând declarații de autentificare, fie dintr-un mesaj care conține un element din care se pot extrage declarațiile de autentificare.

Furnizorii de identitate pot include şi alte tipuri de declarații în răspunsul către inițiator, în funcție de înțelegerile în vigoare şi alte specificații care permit funcționalitatea extinsă.

Fiecare declarație din mesajul `<AuthnResponse>` trebuie semnată individual de către furnizorul de identitate (adică fiecare declarație trebuie să conțină un element Signature care semnează doar declarația în sine). Se recomandă ca semnătura să fie omisă din mesajul `<AuthnResponse>` în sine, dar semnarea nu este interzisă.

### Cerere în plic

Anumite profiluri pot necesita ca cererea să fie introdusă într-un plic. Plicul permite procesarea suplimentară a cererii de către un intermediar, pe drumul între furnizorul de servicii şi cel de identitate. Exemple de intermediari pot fi un agent al utilizatorului sau un proxy. Plicul este îndepărtat la nivelul intermediarului, urmând să trimită la destinatar doar conținutul acestuia, adică un mesaj de tip `<AuthnResponse>`.

Pentru ca îndepărtarea plicului la intermediar să fie efectuată fără probleme, sursa mesajului (în speță furnizorul de servicii) trebuie să se asigure de validitatea conținutului în sine.

### Răspuns în plic

Pentru simetrie, răspunsul furnizorului de identitate poate fi introdus într-un plic, pentru consumul intermediarului (un agent utilizator sau un proxy). Se aplică aceleaşi reguli ca în cazul cererii în plic enunțată în paragraful anterior.

## 4.3. Protocolul Name Registration

În timpul federării, furnizorul de identitate generează un identificator opac care serveşte ca identificator inițial folosit de furnizorul de identitate şi cel de servicii atunci când comunică între ei, referindu-se la client. Identificatorul se numeşte `<IDPProviderNameIdentifier>`.

După federare, furnizorul de servicii poate înregistra un identificator diferit, numit `<SPProviderNameIdentifier>`. Până la momentul în care furnizorul de servicii înregistrează acest identificator, furnizorul de identitate foloseşte `<IDPProviderNameIdentifier>` când se referă la client.

Identificatorul generat de oricare dintre furnizori trebuie să aibă următoarele proprietăți:

- Identificatorul trebuie să fie unic între furnizorii de identitate cu care s-a realizat federarea
- Identificatorul trebuie să fie unic în grupul de identificatori înregistrați de furnizorul de servicii la acest furnizor de identitate

### Cerere

Pentru a înregistra un `<SPProvidedNameIdentifier>` la un furnizor de identitate, furnizorul de servicii trimite un mesaj `<RegisterNameIdentifierRequest>` (vezi figura 13).

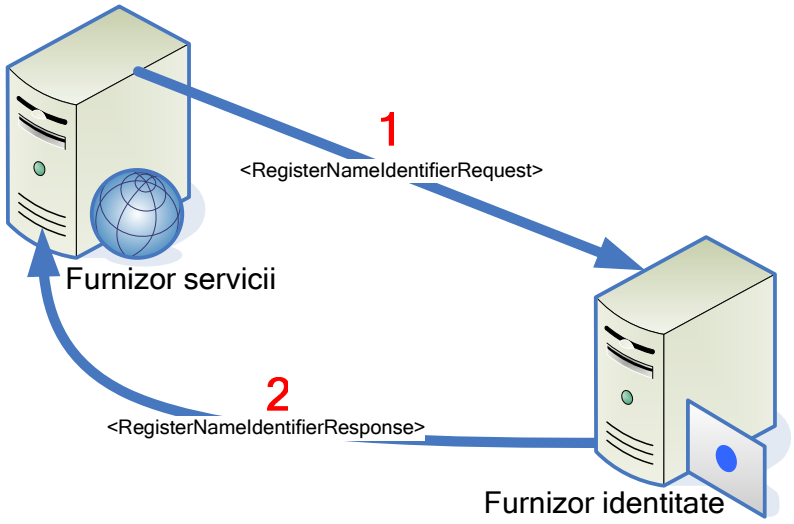


Figura 13. Furnizorul de servicii doreşte înregistrarea unui identificator la furnizorul de identitate

Acelaşi mesaj `<RegisterNameIdentifierRequest>` poate fi trimis de un furnizor de identitate în încercarea de a schimba `<IDPProvidedNameIdentifier>` memorat la furnizorul de servicii (vezi figura 14).

Mesajul `<RegisterNameIdentifierRequest>` fie că este trimis de furnizorul de servicii, fie că este trimis de furnizorul de identitate, trebuie semnat.

### Răspuns

Destinatarul trebuie să răspundă cu un mesaj de confirmare `<RegisterNameIdentifierResponse>` (vezi figura 14). Acest mesaj trebuie să fie semnat.

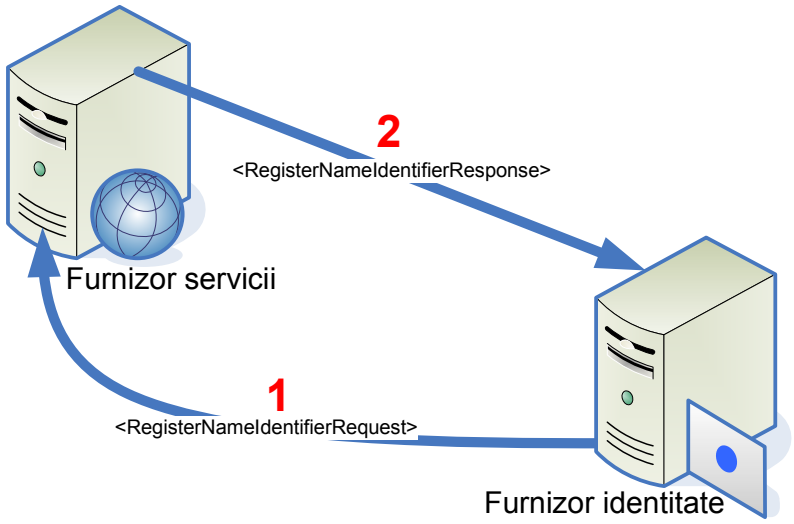


Figura 14. Furnizorul de identitate doreşte schimbarea identificatorului memorat la furnizorul de servicii

## 4.4. Protocolul Federation Termination Notification

Când clientul doreşte să încheie federarea identității între un furnizor de servicii şi un furnizor de identitate de la furnizorul de servicii, acesta trebuie să trimită un mesaj `<FederationTerminationNotification>` către furnizorul de identitate. Prin acesta, furnizorul de servicii declară că nu va mai accepta declarații de autentificare de la furnizorul de identitate pentru clientul în cauză.

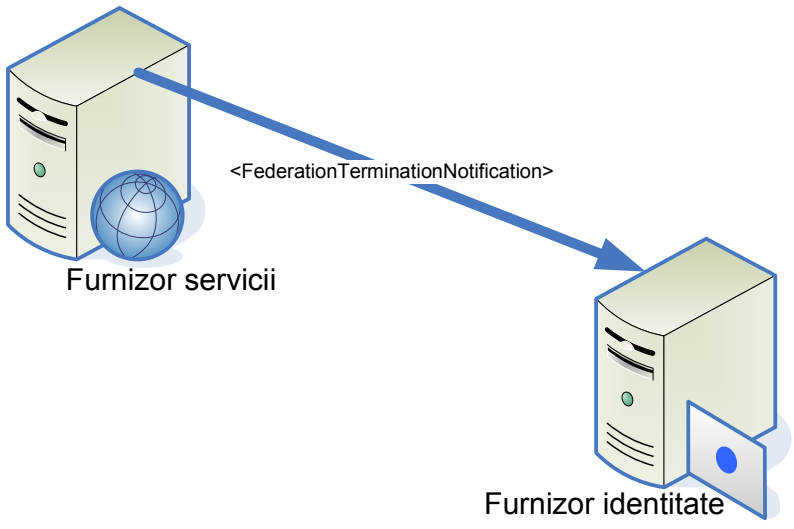


Figura 15. Protocolul de terminare a federării identității

Similar, dacă clientul termină federarea identității de la furnizorul de identitate, acesta trebuie să trimită un mesaj `<FederationTerminationNotification>`

către furnizorul de servicii. În acest caz, furnizorul de identitate declară că nu va mai furniza declarații de autentificare pentru clientul specificat.

Acest mesaj este asincron, într-un singur sens. Entitățile care trimit acest mesaj trebuie să depună un efort rezonabil pentru a livra mesajul. Mesajul trebuie să fie semnat.

## 4.5. Protocolul Single Logout

Protocolul Single Logout este un schimb de mesaje prin care toate sesiunile autentificate de către un furnizor de identitate anume sunt terminate cvasi-simultan. Protocolul se utilizează atunci când clientul doreşte să termine sesiunea de lucru, fie de la nivelul furnizorului de identitate, fie de la cel al furnizorului de servicii.

Când clientul invocă procesul de terminare a sesiunii la furnizorul de servicii, acesta trebuie să trimită un mesaj `<LogoutRequest>` către furnizorul de identitate care a autentificat sesiunea.

Când clientul solicită procesul de terminare a sesiunii la furnizorul de identitate care a autentificat sesiunea sau un furnizor de servicii trimite un astfel de semnal, furnizorul de identitate vizat trebuie să trimită un mesaj `<LogoutRequest>` către fiecare furnizor de servicii căruia i-a trimis declarații de autentificare în sesiunea curentă, în numele clientului, cu excepția furnizorului de servicii care a trimis inițial cererea.

Dacă furnizorul de identitate a generat declarații de autentificare în numele clientului către un furnizor de identitate releu, atunci trebuie să trimită un mesaj `<LogoutRequest>` către acel furnizor, cu excepția cazului în care a primit un astfel de mesaj de la furnizorul în cauză. Vezi figura 16.

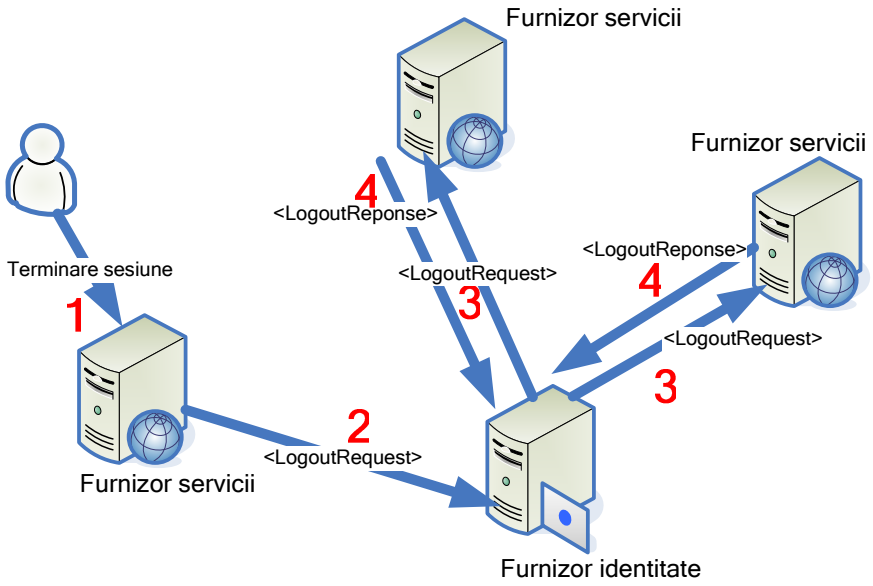


Figura 16. Traseul mesajelor în cazul protocolului Single Logout

### Cerere

Mesajul <LogoutRequest> indică destinatarului că sesiunea utilizatorului s-a terminat. Mesajul trebuie semnat.

### Răspuns

Destinatarul mesajului `<LogoutRequest>` trebuie să trimită confirmarea sub forma mesajului `<LogoutResponse>`. Mesajul trebuie semnat.

## 4.6. Protocolul Name Identifier Mapping

Când un furnizor de servicii cere un identificator pentru un client cu care are o relație de federare a identității, dar care are referințe între furnizorul de identitate şi alt furnizor de servicii, poate folosi acest protocol pentru a obține un astfel de identificator. Acest lucru permite furnizorului de servicii care a făcut cererea să comunice cu un alt furnizor de servicii despre client fără a exista o federare a identității între cei doi. Valoarea rezultată trebuie să fie codificată în aşa fel încât să fie ascunsă tuturor entităților, mai puțin destinația legitimă. Valoarea este o cantitate aleatoare şi de unică folosință, adică un nonce.

La recepționarea mesajului `<NameIdentifierMappingRequest>`, un furnizor de identitate care suportă acest protocol trebuie să răspundă cu un mesaj `<NameIdentifierMappingResponse>`. Vezi figura 17.

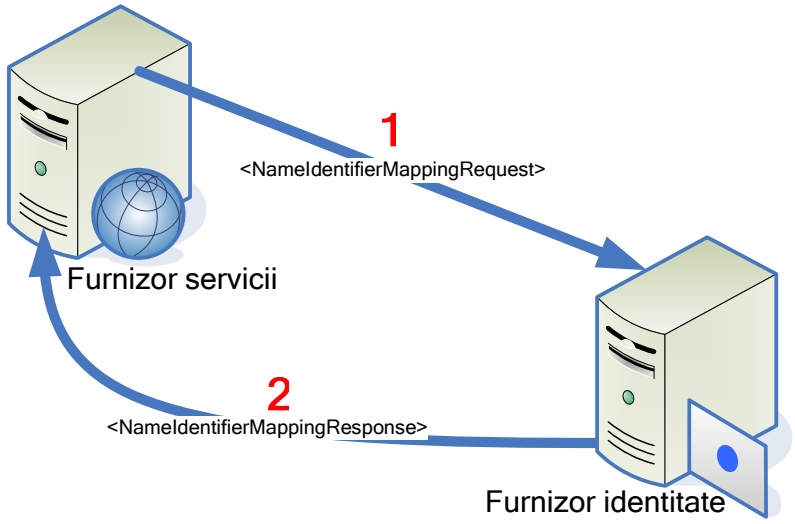


Figura 17. Furnizorul de identitate doreşte schimbarea identificatorului memorat la furnizorul de servicii

### Cerere

Un furnizor de servicii trimite un mesaj `<NameIdentifierMappingRequest>` către un furnizor de identitate care poate genera identificatorul cerut. Mesajul trebuie semnat.

### Răspuns

Furnizorul de identitate destinatar trebuie să răspundă cu un mesaj `<NameIdentifierMappingResponse>`. Mesajul trebuie semnat.

# Capitolul 5

# Atacuri DOS în sisteme Single Sign-On

Atacurile de tip Denial of Service[1] sunt o cauză majoră de operare defectuoasă a sistemelor în Internet şi reprezintă cea mai serioasă ameninţare de astăzi. Primul atac major a îngenunchiat reţeaua Universităţii Minnesota în august 1999, iar 6 luni mai târziu un tânăr canadian a atacat unele dintre cele mai importante site-uri Internet: Yahoo, CNN, Amazon, Buy şi eBay. De atunci atacurile par să fie în creştere.

Din nefericire, utilizatorii sunt mai interesaţi de software-ul cu mai multe facilităţi decât de cel robust, fără erori. În plus, securitatea are preţul său. Software-ul modern cheltuieşte un număr imens de cicluri maşină pentru a desena ferestre tri-dimensionale cu alpha-blending sau alte asemenea îmbunătăţiri vizuale, dar aceste lucruri nu aduc nimic din punct de vedere al funcţionalităţii. Deşi securitatea este o problemă majoră, mulţi utilizatori nu se arată dispuşi să cheltuiască o putere de calcul similară pentru securitate. De asemenea, multor utilizatori nu le pasă dacă sistemul lor este sigur sau poate fi utilizat ca ţintă sau rampă de lansare a unor atacuri de diferite feluri.

Falsul sentiment de securitate este probabil mai rău decât lipsa totală a securităţii. Încă există suficient de mulţi administratori care lasă sistemele lor neprotejate, neaplicând ultimele patch-uri şi neconformându-se procedurilor de bună practică adoptate de fiecare organizaţie. Dacă avem în vedere faptul că numărul de locuinţe, şcoli, biblioteci şi alte locuri publice conectate la Internet a crescut exponenţial în ultima vreme, dimensiunea problemei apare în toată măreţia sa.

Ameninţările de securitate se pot categorisi după cum urmează [MEAD99]:

- Scurgeri de informaţii (confidenţialitate)
- Autentificări nereuşite
- Denial of Service (DOS)

[1] Termenul se traduce aproximativ prin „interzicerea serviciului”, dar am ales să folosesc în această lucrare terminologia originală din limba engleză deoarece este un termen de referinţă în articolele de specialitate.

În timp ce primele două atacuri au fost analizate extensiv în literatura de specialitate, atacurile DOS nu au primit atenția cuvenită până de curând, şi anume după evenimentele din februarie 2000 [CERT00].

## 5.1. Definiția unui atac DOS

Un atac Denial of Service poate lua una din cele două forme posibile. Un atacator poate cauza netransmiterea de către rețea a mesajelor pe care ar trebui să le transmită în mod normal clienților săi. De cealaltă parte se află rețelele care trimit mesaje pe care nu ar trebui să le trimită. De departe cel mai cunoscut atac DOS este cauzarea de trafic fals (inundarea rețelei) în direcția un server particular, lucru care în final va duce la împiedicarea clienților legitimi să obțină serviciul pe care îl cer de la acel server.

Comunitatea Internet cunoaşte o serie întreagă de atacuri DOS, care se pot împărți în două categorii: atacuri prin inundare (flooding) şi atacuri prin pachete modificate.

## 5.2. Cauza atacurilor DOS

O cauză evidentă a atacurilor TCP SYN este că dialogul inițial între părți are loc înaintea unei minime autentificări. Serverul este incapabil de a distinge traficul legitim de cel fals, fapt ce are toate şansele să rămână aşa. Impunerea necesității de a autentifica toate cererile ar însemna un atac DOS în sine, din moment ce serverul ar fi ocupat cu verificarea semnăturilor digital, indiferent dacă acestea sunt sau nu valide. Această nouă cale de atac ar fi la fel de periculoasă ca şi simpla umplere a tabelei TCP.

O cauză mai puțin cunoscută a atacurilor DOS este contabilizarea resurselor, mai precis lipsa acestora. Spatscheck şi Peterson [SPAT99] consideră că sunt trei ingrediente cheie pentru protejarea de atacuri DOS:

- *contabilizarea* tuturor resurselor consumate de un client;
- *detecția* momentului când resursele alocate unui client depăşesc un prag stabilit apriori;
- *constrângerea* – capacitatea de a revendica resursele blocate după detectarea unui atac prin alocarea de resurse minime unui atacator, lucru ce înseamnă automat evitarea unui atac DOS ulterior;

În perioada când Internetul însuşi era proiectat, contabilizarea resurselor era scopul cu prioritatea cea mai mică, lucru ce ne afectează astăzi cel mai mult. În contrast cu rețelele de telefonie omniprezente unde folosirea resurselor era atent controlată, cei care au proiectat rețeaua Internet nu au părut să acorde importanță acestui aspect. Astfel, serverele alocă aceeaşi putere de calcul tuturor cererilor care sosesc la un moment dat ceea ce împiedică o degradare elegantă a performanței în cazul unui atac sau în cazul unei încărcări excesive.

Scenariul anterior este oarecum similar cu mecanismul rudimentar de procesare a pachetelor de intrare datorită arhitecturii bazate pe întreruperi a subsistemului de rețea [DRUS96]. Toate sistemele de operare implementează acest tip de arhitectură care s-a dovedit neadecvat în condiții de încărcare mare. Pachetele de la intrare sunt procesate cu prioritatea maximă pentru ca apoi să fie distruse pentru că nu există nici o aplicație care să le deservească. Această situație se numeşte *receiver livelock.* Mai mult, chiar dacă există o aplicație care să deservească aceste pachete, prioritatea procesului nu este luată în calcul. Astfel, aplicațiile cu prioritate mică primesc aceeaşi cantitate de trafic ca cele cu prioritate mai mare. În lucrarea lor, Druschel şi Banga propun o arhitectură cu de procesare întârziată care se bazează pe demultiplexarea timpurie a pachetelor şi procesarea cu prioritatea destinatarului. Ei susțin că noua arhitectură ar îmbunătăți stabilitatea, justețea

şi capacitatea sistemelor în condiţii de încărcare mare în timp ce performanţa nu ar avea de suferit în condiţii normale.

Crosby şi Wallach [CROS03] au descris o nouă modalitate de atac bazată pe design-ul intrinsec al protocoalelor. Noua clasă de atacuri de bandă îngustă exploatează deficienţele structurilor de date în diferite aplicaţii. Structurile de date folosite în mod uzual au un timp de rulare în cazul mediu mult mai bun decât în cazul cel mai defavorabil. Spre exemplu, atât tabelele de dispersie cât şi arborii binari degenerează în liste înlănţuite atunci când la intrare se prezintă date alese corespunzător. Folosind banda tipică a unui modem, autorii citaţi au adus un server Bro în pragul colapsului; în şase minute de la începutul atacului, serverul ignora 71% din trafic şi consuma întreaga putere de calcul disponibilă.

## 5.3. Ameliorarea efectelor atacurilor DOS

Una dintre cele mai promiţătoare soluţii pentru contracararea efectelor atacurilor de tip *denial of service* este folosirea aşa numitelor client puzzles. Ideea de bază în spatele acestui mecanism este că un eventual client să-şi aloce resursele înaintea serverului ale cărui servicii foloseşte, iar în orice punct al execuţiei protocolului de autentificare, costul rulării pentru client să fie mai mare decât pentru server. Costul pentru client poate fi crescut artificial prin solicitarea rezolvării unui puzzle al cărui grad de dificultate poate fi stabilit cu uşurinţă de către server. În acelaşi timp, verificarea corectitudinii soluţiei nu trebuie să fie o povară pentru server deoarece acest lucru ar anula beneficiile acestei tehnici.

O formă evoluată a tehnologiei client puzzle este cea propusă în [BOCA04], numită threshold puzzle şi este tehnologia preferată pentru ameliorarea atacurilor DOS în sisteme SSO. Vezi anexa 1 pentru mai multe informaţii.

## 5.4. Atacuri DOS în sisteme SSO

Sistemele SSO existente cum ar fi Liberty [LIBE03c], WS-Federation [WSFE03], Shibbolet [SHIB04] sau oricare altul, nu se pot impune ca sisteme complete de management al identităţii într-un mod pur tehnic. Literatura de specialitate menţionează necesitatea coroborării părţii tehnice a implementării cu partea juridică, sub forma unor înţelegeri şi relaţii de afaceri între părţile implicate (vezi modelul din figura 18).

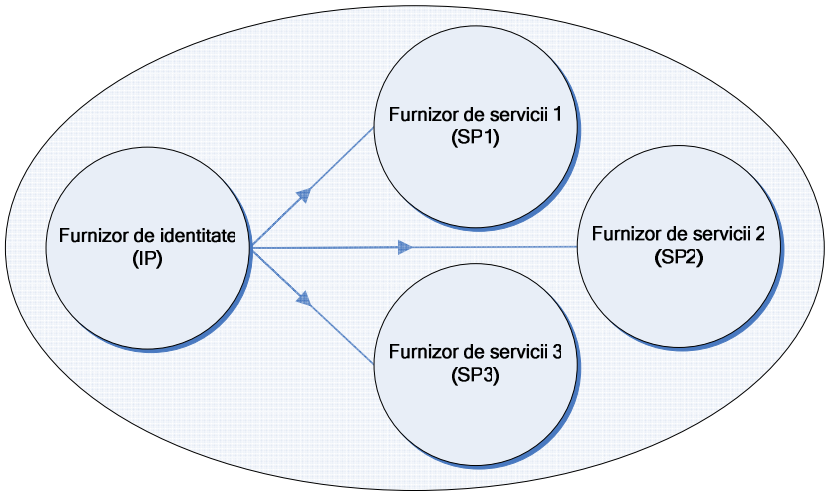


Figura 18. Cercul de încredere bazat exclusiv pe relaţii juridice

Cu toate acestea însă, înțelegerile pot fi încălcate de participanții la cercul de încredere dacă beneficiile sunt mai mari decât pierderile sau dacă unul sau mai mulți furnizori de servicii sunt compromişi, caz în care participanții onești au de suferit.

În cele ce urmează ne vom concentra asupra protocoalelor proiectului Liberty, descris în [LIBE03b] şi vom arăta că acestea sunt vulnerabile la atacuri DOS, neavând o minimă rezistență nativă la acest tip de amenințare.

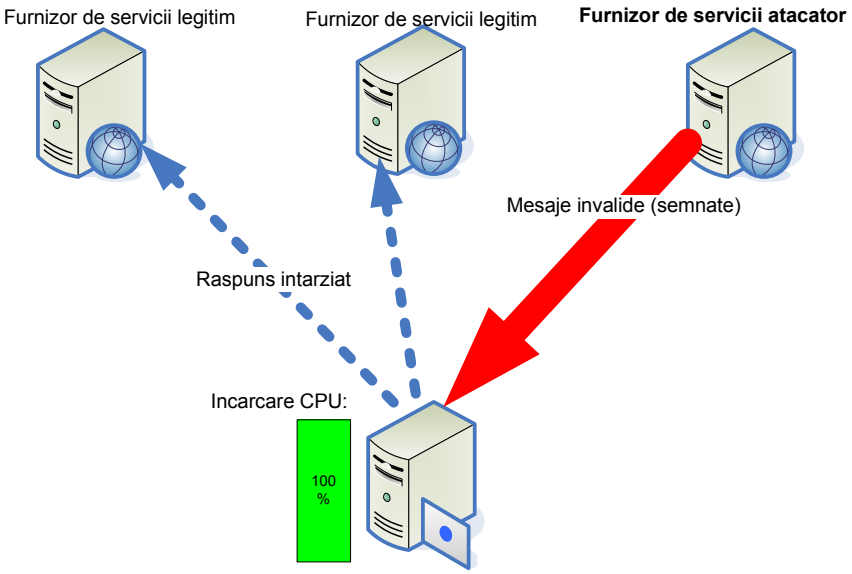


Figura 19. Atacul DOS împotriva unui sistem SSO Liberty (indiferent de protocol)

Ideea din spatele unui atac DOS îndreptat împotriva unui sistem SSO cum este Liberty este dezarmant de simplă (vezi figura 19). Profitând de cerința ca mesajele între entități să fie semnate digital, un furnizor de servicii compromis poate inunda sistemul (în speță furnizorul de identitate) cu mesaje care par legitime, cu semnătură falsă, trimise în cascadă. Nesuspectând nimic, furnizorul de identitate verifică semnăturile digitale ale tuturor mesajelor de intrare, lucru ce duce la epuizarea rapidă a resurselor, ceea ce se traduce fie prin răspunsuri întârziate la cererile furnizorilor legitimi de servicii, fie prin absența cu desăvârşire a acestor răspunsuri adică lipsa serviciului. Aspectele legale ale unui astfel de comportament nu le comentăm, acestea fiind dincolo de scopul acestei lucrări.

Aşa cum am arătat anterior, când s-a pus problema contracarării atacurilor de tip TCP SYN, s-a exclus din start posibilitatea de a semna digital fiecare cerere, deoarece această operație este extrem de scumpă din punct de vedere al timpului de execuție. Un server care îşi pune la dispoziție resursele de calcul fără o minimă autentificare prealabilă este în pericol de a fi suprasaturat cu cereri. În această idee, este foarte curios de aflat de ce dezvoltatorii din Liberty Alliance au trecut cu vederea acest aspect extrem de important. O posibilă explicație ar fi existența cadrului legal în care funcționează sistemul, însă acest lucru nu este suficient pentru protejarea de fapt a participanților onești în sistem.

## 5.5. Propuneri pentru contracararea atacurilor DOS în sisteme SSO

Aşa după cum am văzut deja, protocoalele Liberty sunt vulnerabile la atacuri de tip DOS deoarece resursele părților implicate sunt alocate fără o autentificare minimă prealabilă. Pentru a reduce sau elimina efectele atacurilor DOS asupra sistemelor SSO propun două direcții de acțiune prin care sistemele supuse atacurilor iau măsuri active de apărare. Prima direcție o reprezintă adaptarea tehnologiei threshold puzzles la arhitectura sistemelor SSO prin introducerea a două noi mesaje în cadrul protocolului. A doua direcție vizează relocarea sistemului aflat sub atac la o nouă adresă care se comunică în timp util şi într-un mod securizat clienților legitimi. În cele ce urmează vom detalia cele două propuneri.

## 5.6. Implementarea tehnologiei threshold puzzles în sistemele SSO

În această secțiune propun introducerea unor noi mesaje care permit adoptarea şi implementarea tehnologiei threshold puzzle în fiecare dintre protocoalele Liberty existente.

Ideea în spatele propunerii este următoarea: dacă încărcarea pe furnizorul de identitate este peste un prag stabilit (să spunem 95%), intră în funcțiune mecanismul de protecție prin threshold puzzles. Cererea inițială (prin mesajul `<AuthnRequest>`) este urmată de un mesaj `<PuzzleRequest>` prin care furnizorul de identitate solicită furnizorului de servicii rezolvarea corectă a unei probleme, astfel încât ritmul cererilor sale să scadă. Dacă problema nu este rezolvată corect la recepționarea soluției în mesajul `<PuzzleResponse>`, furnizorul de identitate nu va verifica semnătura digitală a mesajului `<AuthnRequest>`, economisind astfel resurse. Dacă problema a fost rezolvată corect, furnizorul de identitate continuă dialogul obişnuit cu furnizorul de servicii, trimițând mesajul `<AuthnResponse>`.

### 5.6.1. Mesajul <PuzzleRequest>

Mesajul <PuzzleRequest> conține informațiile despre puzzle-ul care trebuie rezolvat de destinatarul mesajului. Acest mesaj **nu trebuie semnat digital**, deoarece această cerință ar anula scopul inițial al folosirii tehnologiei threshold puzzles.

Schema XML pe care o propunem pentru acest mesaj este:

```
<xs:element name="PuzzleRequest" type="PuzzleRequestType"/>
    <xs:complexType name="PuzzleRequestType">
        <xs:complexContent>
            <xs:extension base="samlp:PuzzleChallenge">
                <xs:sequence>
                    <xs:element name="TimeStamp" type="xs:dateTime" minOccurs="0"/>
                    <xs:element name="Difficulty" type="xs:int" minOccurs="0"/>
                    <xs:element name="ServerNonce" type="xs:ulong" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs :complexContent>
        </xs :complexType>
</xs :element>
```

**TimeStamp**

Câmpul TimeStamp indică momentul de timp la care a fost generat puzzle-ul. Pentru a fi acceptată rezolvarea, cantitatea trebuie să conțină un timp recent, în acord cu specificațiile de implementare a tehnologiei threshold puzzle.

**Difficulty**

Câmpul Difficulty indică dificultatea puzzle-ului dictată de către entitatea care l-a generat. Valorile sunt pe o scară exponențială care variază de la 0 (nu este necesară depunerea nici unui efort) şi 128 (imposibil).

**ServerNonce**

Cantitatea ServerNonce este o valoare aleatoare de 64 de biți generată aleatoriu.

### 5.6.2. Mesajul <PuzzleResponse>

Mesajul <PuzzleResponse> este trimis de către entitatea care rezolvă puzzle-ul către entitatea care l-a generat, pentru verificare. Mesajul **nu trebuie semnat digital**. Schema XML pe care o propunem este următoarea:

```
<xs:element name="PuzzleResponse" type="PuzzleResponseType"/>
    <xs:complexType name="PuzzleResponseType">
        <xs:complexContent>
            <xs:extension base="samlp:PuzzleSolution">
                <xs:sequence>
                    <xs:element name="ClientID" type="xs:string" minOccurs="0"/>
                    <xs:element name="ServerNonce" type="xs:ulong" minOccurs="0"/>
                    <xs:element name="ClientNonce" type="xs:uint" minOccurs="0"/>
                    <xs:element name="Solution" type="xs:ulong" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
       </xs:complexType>
</xs:element>
```

**ClientID**

Câmpul ClientID conține identificatorul clientului (entitatea care primeşte puzzle-ul spre rezolvare) şi este de regulă adresa IP sau un alt identificator generat după reguli stabilite prin specificațiile de implementare.

**ServerNonce**

Cantitatea ServerNonce este o valoare aleatoare de 64 de biți generată aleatoriu.

**ClientNonce**

Cantitatea ClientNonce este o valoare aleatoare de 64 de biți generată aleatoriu.

**Solution**

Câmpul Solution conține rezolvarea puzzle-ului.

### 5.6.3. Aspecte de implementare

Tehnologia threshold puzzles se poate adapta uşor la fiecare dintre protocoalele proiectului Liberty. Modificarea trebuie să se facă după următorul algoritm:

- La recepționarea unui mesaj care implică verificarea unei semnături digitale, furnizorul de identitate ia decizia daca activează sau nu mecanismul threshold puzzles în funcție de încărcarea curentă.
- Dacă încărcarea curentă trece de un prag stabilit, se salvează contextul curent al mesajului şi se răspunde cu mesajul <PuzzleRequest>, apoi se aşteaptă răspunsul.
- La sosirea răspunsului sub forma mesajului <PuzzleResponse> se verifică validitatea soluției puzzle-ului, iar dacă aceasta este corectă se trece la compunerea răspunsului pentru mesajul inițial.
- Dacă soluția puzzle-ului este incorectă, contextul mesajului inițial se distruge.

Protocolul de terminare a federării prezintă o particularitate. Acest protocol este compus dintr-un singur mesaj asincron, care anunță furnizorul de identitate asupra terminării federării. Deoarece nu se așteaptă nici un răspuns, conform designului inițial este posibil ca emițătorul mesajului să nu mai aibă cunoștință de contextul în care a trimis mesajul inițial în cazul în care receptorul solicită rezolvarea unui puzzle. Se impune deci adăugarea unui mesaj de confirmare a defederării la sfârșitul protocolului sau un folosirea unui timp de gardă după care se consideră implicit că defederarea a avut loc.

Detalii despre implementarea tehnologiei threshold puzzle sunt menționate în anexa 1.

## 5.7. Relocarea furnizorilor de identitate

Sistemele critice care trebuie să își mențină serviciile la un standard ridicat pot adopta o tehnică suplimentară de apărare împotriva atacurilor DOS. Deşi tehnologia threshold puzzles este în cele mai multe cazuri eficientă, activarea ei este în măsură să încetinească întrucâtva clienții furnizorilor de identitate. De aceea, propunem o metodă de apărare prin schimbarea adresei furnizorului de identitate, sau relocare.

Principiul relocării este simplu: pentru a putea ataca un sistem, atacatorul trebuie să îi cunoască locația sau altfel spus adresa IP. Pentru a se apăra de atac, victima poate decide să își schimbe adresa IP dintr-o plajă care îi este disponibilă (vezi figura 20). Clienții legitimi vor afla noua adresă prin intermediul unui mesaj de notificare.

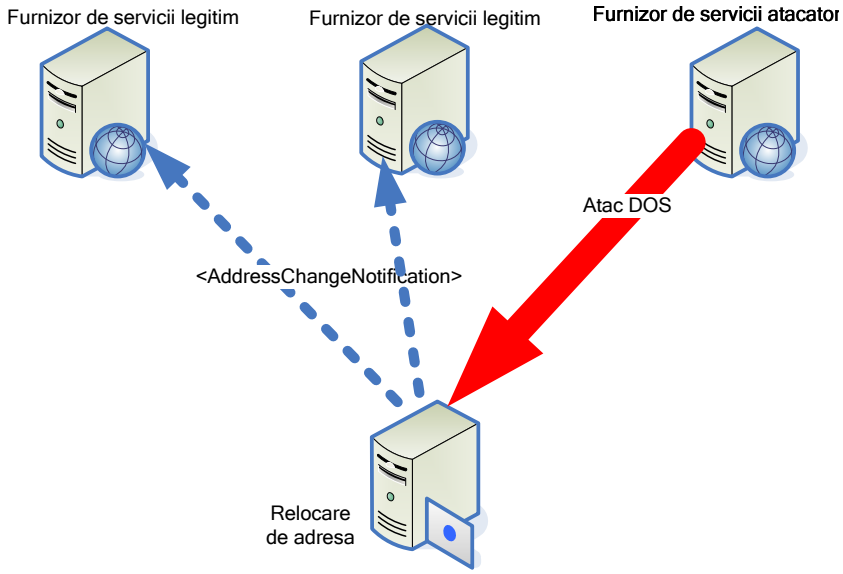


Figura 20. Notificarea schimbării locației unui furnizor de identitate

Furnizorul de identitate va primi de acum înainte mesajele la noua locație. Atacatorului nu i se transmite noua locație, acesta fiind înscris pe o lista neagră (vezi figura 21). În cazul în care la momentul relocării au existat furnizori de servicii legitimi care din diverse motive nu au primit notificarea cu noua adresă a furnizorului de identitate (spre exemplu au fost off-line), aceştia vor afla noua locație prin intermediul mesajelor de notificare baliză pe care furnizorul de identitate le trimite la intervale regulate doar furnizorilor de servicii înregistrați. Mesajele baliză încetează în momentul în care toți furnizorii de servicii care nu sunt pe lista neagră au trimis cel puțin un mesaj valid. De remarcat că mesajele baliză se trimit doar anumitor entități, deci nu sunt mesaje broadcast care ar putea fi recepționate şi de entitățile compromise.

### 5.7.1. Mesajul <IdentityProviderRelocationNotification>

Mesajul <IdentityProviderRelocationNotification> se trimite în momentul în care furnizorul de identitate decide să îşi schimbe locaţia (adresa IP) pentru a se proteja fizic de atacuri. Schema XML propusă a mesajului este:

```
<xs:element name="IdentityProviderRelocationNotification"
type="IdentityProviderRelocationNotificationType"/>
    <xs:complexType name="IdentityProviderRelocationNotificationType">
        <xs:complexContent>
            <xs:extension base="samlp:RelocationNotification">
                <xs:sequence>
                    <xs:element name="RelocatedAddress" type="xs:string" minOccurs="0"/>
                    <xs:element name="RelocationEffective" type="xs:dateTime"
               minOccurs="0"/>
                    <xs:element ref="ProviderID"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
       </xs:complexType>
</xs:element>
```

**RelocatedAddress -** Reprezintă noua adresa la care se va reloca furnizorul de identitate.

**RelocationEffective** – Reprezintă momentul de timp la care relocarea devine efectivă, pentru a permite sincronizarea furnizorilor de servicii

**ProviderID** – Reprezintă identitatea entităţii care a trimis mesajul

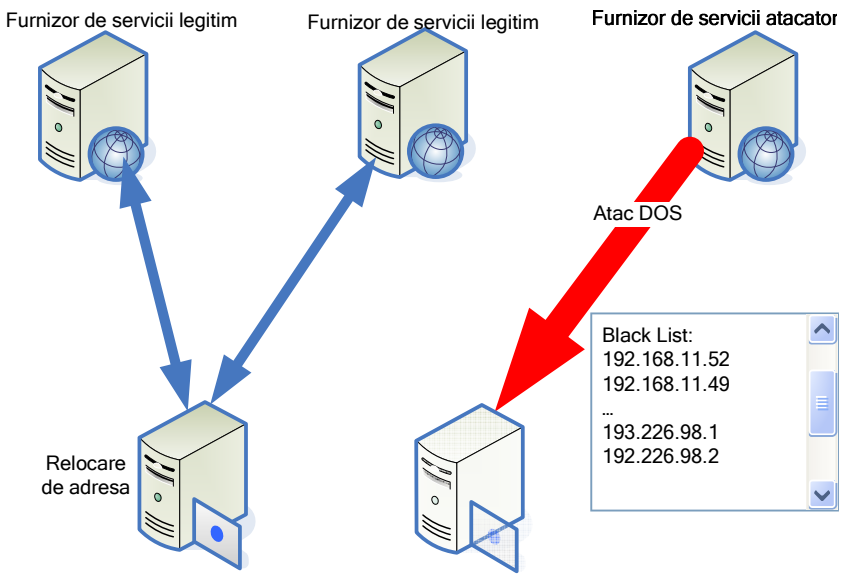


Figura 21. Atacatorul nu mai poate comunica cu furnizorul de identitate întrucât acest şi-a relocat adresa

### 5.7.2. Mesajul <IdentityProviderRelocationAcknowledge>

Mesajul <IdentityProviderRelocationAcknowledge> se trimite furnizorului de identitate la noua sa locaţie, confirmându-i schimbarea de locaţie. Schema XML propusă a mesajului este:

```
<xs:element name="IdentityProviderRelocationAcknowledge"
type="IdentityProviderRelocationAcknowledgeType"/>
    <xs:complexType name="IdentityProviderRelocationAcknowledgeType">
```

```
        <xs:complexContent>
            <xs:extension base="samlp:RelocationAcknowledge">
                <xs:sequence>
                    <xs:element ref="ProviderID"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
       </xs:complexType>
</xs:element>
```

**ProviderID** – Reprezintă identitatea entității care a trimis mesajul

### 5.7.3. Mesajul <RelocationBeacon>

Mesajul <RelocationBeacon> se trimite furnizorului de identitate la noua sa locație, confirmându-i schimbarea de locație. Schema XML propusă a mesajului este:

```
<xs:element name="RelocationBeacon" type="RelocationBeaconType"/>
    <xs:complexType name="RelocationBeaconType">
        <xs:complexContent>
            <xs:extension base="samlp:RelocationBeacon">
                <xs:sequence>
                    <xs:element name="RelocatedAddress" type="xs:string" minOccurs="0"/>
                    <xs:element ref="ProviderID"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
       </xs:complexType>
</xs:element>
```

**RelocatedAddress -** Reprezintă noua adresă la care furnizorul de identitate s-a relocat.

**ProviderID** – Reprezintă identitatea entității care a trimis mesajul

### 5.7.4. Mesajul <RelocationBeaconAcknowledge>

Mesajul <RelocationBeaconAcknowledge> se trimite furnizorului de identitate la noua sa locație, confirmându-i schimbarea de locație. Schema XML propusă a mesajului este:

```
<xs:element name="RelocationBeaconAcknowledge" type="RelocationBeaconType"/>
    <xs:complexType name="RelocationBeaconType">
        <xs:complexContent>
            <xs:extension base="samlp:BeaconAcknowledge">
                <xs:sequence>
                    <xs:element ref="ProviderID"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
       </xs:complexType>
</xs:element>
```

**ProviderID** – Reprezintă identitatea entității care a trimis mesajul

# Capitolul 6

# Concluzii

## 6.1. Concluzii

În cadrul lucrării a fost abordată problematica protocoalelor de autentificare Single Sign-On (SSO) şi federare a identităţii din perspectiva vulnerabilităţii acestora la atacuri de tip DOS (Denial-of-Service). În capitolul 1 s-a stabilit contextul în care protocoalele SSO au apărut (furtul de identitate tot mai des întâlnit în mediul electronic), precum şi beneficiile pentru companii şi clienţi. Tot în acest capitol s-au prezentat avantajele sistemelor SSO, după cum urmează:

- Securitate şi confidenţialitate sporite în tranzacţiile dintre client şi furnizorii de servicii şi cei de identitate
- Nu există un singur punct vulnerabil, adică în orice punct există o cantitate limitată de informaţii
- Acces la atributele clientului bazat pe listă de permisiuni.
- Actualizări ale documentaţiei şi protocoalelor pentru a ţine pasul cu atacurile şi vulnerabilităţile descoperite
- Răspuns imediat la incidente

Deosebim două tipuri de sisteme SSO pe care le-am prezentat pe scurt în capitolul 2. Primul dintre acestea, numit „pseudo-SSO", funcţionează ca un strat intermediar între utilizator şi aplicaţiile pe care acesta le accesează. Autentificarea iniţială (numită *autentificare primară*) se face între utilizator şi sistemul SSO. La fiecare acces la o aplicaţie care necesită credenţialele utilizatorului, sistemul SSO intervine şi efectuează paşii necesari pentru ca autentificarea să fie reuşită. Pentru fiecare acces la aplicaţie se efectuează o autentificare într-o modalitate transparentă pentru utilizator. Aplicaţia poate să nu fie informată că are loc o autentificare comandată de către sistemul SSO.

Cel de-al doilea tip de sisteme SSO este cel numit „true-SSO". În acest caz, autentificarea primară se realizează către o componentă numită *Authentication Service Provider (ASP)*. Unele lucrări numesc această componentă *Identity Service Provider (ISP)*. Pentru a realiza SSO, ASP trebuie să aibă stabilită o relaţie cu fiecare furnizor de servicii (SP), relaţie care de regulă este consfinţită printr-un contract legal. De asemenea trebuie să existe un canal securizat de comunicaţie între ASP şi SP. Odată ce autentificarea primară a avut loc, la cererea furnizorilor de servicii (SP), ASP-ul

realizează autentificarea prin intermediul unui protocol dedicat, folosind aşa-numitele *authentication assertions*. Acestea conţin identitatea utilizatorului şi starea sa aşa cum este cunoscută de către ASP şi trebuie transmise în mod securizat, în funcţie de specificul implementării.

În funcţie de localizarea componentei de autentificare, deosebim următoarele tipuri de sisteme SSO:

- Sisteme pseudo-SSO locale
- Sisteme pseudo-SSO bazate pe proxy
- Sisteme SSO locale
- Sisteme SSO bazate pe proxy

Capitolul 3 face o introducere în conceptul de federare a identităţii, conceptul de single sign-on şi prezintă scenarii de utilizare din punctul de vedere al utilizatorului. Federarea identităţii reprezintă crearea unei legături logice între insulele de identitate aflate la diverşi furnizori de servicii pentru îmbunătăţirea experienţei utilizatorului în ceea ce priveşte comoditatea autentificării. Single Sign-On este procesul prin care autentificarea primară efectuată la unul dintre membrii cercului de încredere este „replicată" automat fiecărui furnizor de servicii. Acest proces se bazează pe consimţământul expres al utilizatorului, dat în pasul anterior, federarea identităţii.

Misiunea unui sistem de federare a identităţii cuprinde:

- Permite utilizatorilor să-şi protejeze identitatea în reţea
- Permite companiilor să menţină contactul cu clienţii fără intervenţia unui terţ
- Furnizarea unui sistem single sign-on care include autentificare şi autorizare descentralizate, de la mai mulţi furnizori de identitate
- Crearea unei infrastructuri care să permită interconectarea dispozitivelor de reţea actuale şi viitoare

În capitolul 4 am prezentat succint protocoalele aflate la baza Liberty Alliance:

- Protocolul Single Sign-On and Federation
- Protocolul Name Registration
- Protocolul Federation Termination Notification
- Protocolul Single Logout
- Protocolul Name Identifier Mapping

Capitolul 5 face o prezentare a atacurilor DOS, a cauzelor apariţiei unor astfel de atacuri şi modalităţilor de ameliorare ale efectelor. În lucrarea de faţă se vorbeşte despre atacuri DOS asupra sistemelor SSO pentru prima dată şi se arată că protocoalele Liberty nu au o protecţie nativă la acest fel de atacuri. Tot în cadrul acestui capitol se face o propunere de utilizare a tehnologiei threshold puzzles şi modificarea schimbului de mesaje în protocoalele Liberty pentru a beneficia de protecţie la atacuri DOS. Mai mult, pentru sistemele speciale care au nevoie de timpi de răspuns foarte mici, protecţia poate să includă chiar relocarea anumitor servicii, metodă pentru care de asemenea se propun câteva mesaje noi.

Pentru a întregi tabloul, anexa 1 face o prezentare detaliată a conceptului de puzzle, care trebuie să aibă următoarele proprietăţi:

- Crearea unei puzzle şi verificarea soluţiei nu necesită resurse importante din partea serverului.
- Costul rezolvării puzzle-ului este uşor a fi modificat de la 0 la infinit.
- Puzzle-ul poate fi rezolvat pe majoritatea platformelor hardware.
- Precalcularea soluţiei puzzle-ului este imposibilă.

- În timp ce clientul rezolvă puzzle-ul, serverul nu trebuie să memoreze soluția sau alte informații specifice clientului.
- Acelaşi puzzle poate fi distribuit mai multor clienți. Cunoscând soluțiile calculate de unul sau mai mulți clienți nu ajută în calcularea unei noi soluții.
- Un client poate reutiliza un puzzle prin crearea mai multor instanțe ale sale.

Puzzle-ul propus în mod uzual este inversiunea prin forță brută a unei funcții de dispersie cum ar fi MD5 sau SHA1.

## 6.2. Rezumatul contribuțiilor

Referatul de față avansează următoarele idei principale:

- Sistemele Single Sign-On (SSO) au fost proiectate fără a lua în calcul factorii care duc la existența atacurilor DOS, aceasta fiind o importantă breşă de securitate.
- Schimbul de mesaje în cadrul protocoalelor Liberty trebuie să se realizeze securizat prin semnătură digitală, fără o minimă autentificare prealabilă, fapt ce duce la posibilitatea unui atac prin epuizarea resurselor de calcul.
- În caz de atac, sistemul SSO este complet vulnerabil, lucru cu atât mai grav cu cât sistemul poate avea cerința de disponibilitate maximă.

Având la bază aceste principii, lucrarea aduce următoarele contribuții în domeniul securizării sistemelor Single Sign-On:

- Avertizarea asupra faptului că sistemele Single Sign-On în general sunt complet vulnerabile la atacuri DOS, luându-se ca exemplu concret sistemul Liberty.
- Propunerea de a se folosi tehnologia „threshold puzzles" pentru ameliorarea efectelor unui atac DOS.
- Definirea a două mesaje noi (<PuzzleRequest> şi <PuzzleResponse>) şi includerea lor în cursul normal al fiecărui protocol Liberty.
- Propunerea ca sistemele SSO speciale care au ca cerință o disponibilitate ridicată să facă uz de relocarea serviciului aflat sub atac şi propunerea unui schimb de mesaje în acest scop.
- Definirea a patru noi mesaje (<IdentityProviderRelocationNotification>, < IdentityProviderRelocationAcknowledge>, <RelocationBeacon>, <RelocationBeaconAcknowledge>) pentru realizarea relocării serviciului aflat sub atac, atunci când tehnologia threshold puzzles începe să afecteze calitatea serviciului.

## 6.3. Perspective de cercetare şi dezvoltare

Preocupările curente şi de viitor din cadrul activității de doctorat cuprind următoarele subiecte:

- Extinderea analizei vulnerabilității la atacuri DOS şi la alte protocoale SSO existente (WS-Federation, Shibboleth, etc.)
- Definirea unui cadru teoretic de protecție la atacuri DOS care să cuprindă propunerile din această lucrare
- Implementarea practică a propunerilor şi argumentarea de fapt a beneficiilor aduse de modificările protocoalelor

# Bibliografie

[BACK01] Adam Beck, Ulf Moller, Anton Stiglic – *Traffic analysis attacks and trade-offs in anonymity providing systems,* Information Hiding, 4th International Workshop, IHW 2001, volume 2137 of Lecture Notes in Computer Science, pages 245-257, Springer-Verlag, Berlin 2001

[BIRO02] Biron, P. V., Malhotra, A. – *XML Schema Part 2: Datatypes*, Recommendation, World Wide Web Consortium, http://www.w3.org/TR/xmlschema-2, mai 2002

[BOCA04] Valer Bocan – *Threshold Puzzles: The Evolution of DOS-resistant Authentication*, Periodica Politehnica, Transactions on Automatic Control and Computer Science, Vol. 49 (63), CONTI 2004

[CANT03] Cantor, Scott, Kemp, John – *Liberty ID-FF Bindings and Profiles Specification, Version 1.2,* Liberty Alliace Project, November 2003, http://www.projectliberty.org/specs

[CERT00] Computer Emergency Response Team - *CERT advisory CA-2000.01 Denial of service developments*, 2000 (http://www.cert.org/advisories/CA-2000-01.html)

[CHAU81] David L. Chaum – *Untraceable electronic mail, return addresses and digital pseudonims*, Communications of the ACM, 24(2):84-90, 1981

[CLER02] Jan de Clercq – *Single Sign-On Architectures*, Infrastructure Security, International Conference, InfraSec 2002, Bristol, UK, 2002

[CROS03] Scott A. Crosby, Dan S. Wallach – *Denial of Service via Algorithmic Complexity Attacks*, Proceedings of the12th USENIX Security Symposium, 2003

[DEAN01] Drew Dean, Adam Stubblefield – *Using Client Puzzles to Protect TSL*, http://www.csl.sri.com/users/ddean/papers/usenix01b.pdf, Proceedings of the10th USENIX Security Symposium, 2001

[DRUS96] Peter Druschel, Gaugrav Banga – *Lazy receiver processing (LRP): a network subsystem architecture for server systems*, Proceedings of the 2nd USENIX Symposium on OSDI, Seattle, 1996

[DWOR92] Cynthia Dwork, Moni Naor – *Pricing via Processing or Combating Junk Mail*, Proceedings of CRYPTO '92, Springer Verlag, 1992

[GOLD99] David M. Goldschlag, Michael G. Reed, Paul F. Syverson – *Onion routing for anonymous and private Internet connections*, Communications of the ACM, 42(2):84-88, January 1999

[JUEL99] Ari Juels, John Brainard – *Client puzzles: A cryptographic defense against connection depletion attacks*, Proceedings of the NDSS 1999

[LIBE03a] Liberty Alliance Project – *Liberty ID-FF Architecture Overview 1.2*, http://www.projectliberty.org

[LIBE03b] Liberty Alliance Project – *Liberty ID-FF Protocols and Schema Specification 1.2,* http://www.projectliberty.org

[LIBE03c] Liberty Alliance Project – *Liberty Specs Tutorial*, http://www.projectliberty.org

[LIBE04a] Liberty Alliance Project – *Whitepaper on Liberty Protocol and Identity Theft*, 20 februarie 2004, http://www.projectliberty.org

[MALE03] Maler, E., Mishra, P., Philpott, R. – *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) 1.1*, OASIS Committee Specification, Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

[MEAD99] Catherine Meadows – *A formal framework and evaluation method for network denial of service*, Proceeding of the 1999 IEEE Computer Security Foundations Workshop, Mordano, Italy, 1999

[MERK78] R. C. Merkle – *Secure Communications Over Insecure Channels*, Communications of the ACM, 1978

[PASH03] Andreas Pashalidis, Chris J. Mitchell – *A Taxonomy of Single Sign-On Systems*, 2003, Royal Holloway, University of London

[PFIT01] Andreas Pfitzmann, Marit Köhntopp – *Anoymity, unobservability and pseudonimity – a proposal for terminology*, Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, number 2009 in Lecture Notes in Computer Science, pages 141 – 160, Springer-Verlag, Berlin 2001

[RIVE96] Ronald R. Rivest, Adi Shamir, David A. Wagner – *Time-lock Puzzles and Timed-release Cryptography*, 1996, http://lcs.mit.edu/~rivest/RivestShamirWagner-timelock.pdf

[RFC3280] Housley R. – *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,* RFC 3280, The Internet Engineering Task Force, http://www.rfc-editor.org/rfc/rfc3280.txt

[SHIB04] Shibboleth Architecture, Protocols and Profiles, Working Draft 02, 22 September 2004, http://shibboleth.internet2.edu

[SHON01] Shon Harris – *DOS Defense*, Information Security Magazine, 2001

[SPAT99] Oliver Spatscheck, Larry Peterson – *Defending against denial of service in Scout*, Proceedings of 3rd USENIX/ACM Symposium on OSDI, p. 59-72, 1999

[TUOM00] Tuomas Aura, Pekka Nikander, Jussipekka Leiwo – *DOS-resistant authentication with client-puzzles*, Proceeding of the Cambridge Security Protocols Workshop 2000, LNCS, Cambridge, UK, 2000

[WSFE03] IBM Corporation, Microsoft Corporation, BEA Systems, RSA Security, Verisifg - *Web Services Federation Language (WS-Federation)*, July 2003, http://www.ibm.com/developerworks/library/ws-fed/

[WWW1] Anonimizer – www.anonymizer.com

[WWW2] WebSecure - http://www.freedom.net/products/websecure

[WWW3] JAP Anonymity and Privacy – http://anon.inf.tu-dresden.de

[WWW4] SETI @home Program, http://setiathome.ssl.berkely.edu

[WWW5] The Distributed.net Organization, http://www.distributed.net

[WWWC02] World Wide Web Consortium – *The Platform for Privacy Preferences 1.0 (P3P 1.0) Specification, April 2002*

# Anexa 1

# Puzzles

Una dintre cele mai promițătoare soluții pentru contracararea efectelor atacurilor de tip *denial of service* este folosirea aşa numitelor client puzzles. Ideea de bază în spatele acestui mecanism este ca un eventual client să-şi aloce resursele înaintea serverului ale cărui servicii foloseşte, iar în orice punct al execuției protocolului de autentificare, costul rulării pentru client să fie mai mare decât pentru server. Costul pentru client poate fi crescut artificial prin solicitarea rezolvării unui puzzle al cărui grad de dificultate poate fi stabilit cu uşurință de către server. În acelaşi timp, verificarea corectitudinii soluției nu trebuie să fie o povară pentru server deoarece acest lucru ar anula beneficiile acestei tehnici.

Într-o lucrare din anul 1978, Merkle [MERK78] a fost primul care a venit cu ideea de puzzle-uri criptografice dar a aplicat ideea doar pentru schimbul de chei şi nu în autentificare. Client puzzle-urile au fost aplicate în inundarea TCP SYN de către Juels şi Brainard [JUEL99] care menționează că SSL are aceeaşi slăbiciune şi dau o demonstrație riguroasă a caracteristicilor de securitate. Aura, Nikander şi Leiwo au aplicat puzzle-urile în protocoalele de autentificare în general [TUOM00] iar Dwork şi Naor [DWOR92] au propus măsuri de reglementare pentru mesajele nesolicitate. Rivest, Shamir şi Wagner [RIVE96] discută o problemă conexă a criptografiei blocată în timp.

## A1.1. Descriere

Înainte să aloce resurse pentru deservirea unei conexiuni serverul trebuie să se asigure că pe întreaga durată a execuției protocolului costul pentru client este mai mare decât pentru server. Costul pentru client se poate mări artificial prin solicitarea rezolvării unui puzzle care trebuie să aibă anumite proprietăți., după cum urmează [TUOM00, SHON01]:

- Crearea unui puzzle şi verificarea soluției nu necesită resurse importante din partea serverului.
- Costul rezolvării puzzle-ului este uşor de a fi modificat de la 0 la infinit.
- Puzzle-ul poate fi rezolvat pe majoritatea platformelor hardware.
- Precalcularea soluției puzzle-ului este imposibilă.

- În timp ce clientul rezolvă puzzle-ul, serverul nu trebuie să memoreze soluția sau alte informații specifice clientului.
- Același puzzle poate fi distribuit mai multor clienți. Cunoscând soluțiile calculate de unul sau mai mulți clienți nu ajută în calcularea unei noi soluții.
- Un client poate reutiliza un puzzle prin crearea mai multor instanțe ale sale.

Puzzle-ul propus în mod uzual este inversiunea prin forță brută a unei funcții de dispersie cu sens unic cum ar fi MD5 sau SHA1. Această propunere este potrivită din moment ce funcțiile hash sunt uşor de implementat pe o mare varietate de platforme hardware, iar testarea succesivă a tuturor intrărilor este cel mai probabil cea mai eficientă metodă de a inversa o funcție de dispersie. Principiul general este arătat în figura 22.

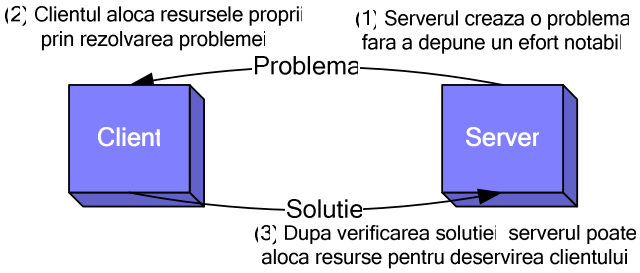


Figura 22. Principiul *client puzzle*

## A1.2. Crearea unui puzzle

Periodic (să spunem o dată la câteva minute), serverul generează o valoare aleatoare $\mathbf{N_S}$. Pentru a împiedica atacurile prin ghicire, valoarea trebuie să aibă o entropie de 64 de biți şi să nu fie o valoare predictibilă în vreun fel, ca de exemplu amprenta de timp. Această entropie ar trebui să fie suficientă pentru a împiedica un atacator să calculeze perechi «$N_S$, rezultat». Potrivirile ocazionale rezultate din atacuri de tip „zi de naştere" nu au efecte prea importante în acest caz.

Serverul trebuie să decidă de asemenea şi nivelul de dificultate **k** al puzzle-ului pe baza unui cumul de măsurători ale condițiilor curente. În rezumat, puzzle-ul trimis clienților arată astfel:

$$\mathbf{« N_S, k »}$$

- $N_S$ – valoarea aleatoare generată de server (de obicei o cantitate cu mărimea de 64 biți)
- k – nivelul de dificultate a puzzle-ului

## A1.3. Rezolvarea puzzle-ului

Pentru a rezolva puzzle-ul, clientul generează la rândul său o valoare aleatoare $\mathbf{N_C}$. Scopul acestei valori este dublu. În primul rând, dacă clientul refoloseşte valoarea $\mathbf{N_S}$ generată de server, poate construi un nou puzzle prin generarea unei noi valori $N_C$. În al doilea rând, fără această valoare, un atacator ar putea calcula puzzle-ul înaintea clientului şi ar trimite rezultatul serverului. 24 de biți de entropie ar trebui să fie îndeajuns pentru a împiedica atacatorul de a epuiza întreaga plajă de valori ale $N_c$ dat fiind faptul că $N_S$ se schimbă frecvent.

Clientul trebuie să aplice în mod repetat o funcție de dispersie unei cantități iar puzzle-ul se consideră rezolvat când primii k biți ai cantității Y sunt egali cu 0.

$$\mathbf{h(C, N_S, N_C, X) = Y}$$

1. h – funcție criptografică de dispersie, cum ar fi MD5 or SHA
2. C – identitatea clientului
3. $N_S$ – valoarea aleatoare generată de server
4. $N_C$ – valoarea aleatoare generată de client
5. X – soluția puzzle-ului

Din moment ce serverul schimbă $N_S$ periodic, atât timp cât acest $N_S$ se consideră recent, serverul trebuie să mențină o listă de perechi $N_S$ – $N_C$ în aşa fel soluțiile vechi să nu poată fi refolosite.

Din moment ce nu se cunoaşte nici o metodă ocolitoare pentru a determina X, singura posibilitate este ca această cantitate să se determine secvențial, adică prin forță brută. Nivelul de dificultate k (adică numărul de biți 0 de la începutul lui Y) dictează cât de mult va lua rezolvarea puzzle-ului. Dacă k este egal cu 0 nu este necesar nici un efort, iar dacă k este egal cu 128 (pentru MD5) sau 192 (pentru SHA), clientul trebuie să inverseze o întreagă funcție de dispersie, lucru imposibil computațional.

## A1.4. Dificultatea puzzle-ului

Parametrul k reprezintă dificultatea puzzle-ului. Sarcina de a stabili această valoare este destul de dificilă deoarece nu există nici o metrică ce s-ar putea folosi într-o implementare reală. În conformitate cu [DEAN01], cea mai potrivită abordare ar fi luarea în calcul a numărului de operații RSA alocate deja. Încărcarea curentă a procesorului şi numărul de conexiuni la intrare sunt metrici care depind de un număr de factori care le fac nepotrivite în implementările reale.

Din nefericire, timpul de rezolvare în funcție de dificultatea puzzle-ului urmează o curbă exponențială, aplicabilitatea practică fiind astfel limitată. Pentru a rezolva un puzzle de dificultate k, clientul trebuie să efectueze în medie aproximativ $2^k - 1$ operații. În [TUOM00], Aura, Nikkander şi Leiwo susțin că valorile rezonabile pentru k sunt între 0 şi 64. Prin experiment am aflat că plaja rezonabilă este mult mai îngustă, iar pentru valori mici ale nivelului de dificultate, timpul necesar pentru rezolvarea nivelului k poate fi uneori mai mare decât pentru nivelul k + 1.

Astăzi (începutul anului 2004), un client web tipic este capabil de aproximativ 4500 – 5000 MIPS ceea ce duce la 0,02 milisecunde per operație criptografică. Timpii de execuție pentru fiecare nivel de dificultate sunt dați în tabelul 1:

Tabelul 1 Timpul mediu de execuție pentru diverse niveluri de dificultate

| Nivel de dificultate | Timp de execuție (ms) | Nivel de dificultate | Timp de execuție (ms) |
|---|---|---|---|
| 1 | 0 | 11 | 37 |
| 2 | 0 | 12 | 155 |
| 3 | 0 | 13 | 245 |
| 4 | 1 | 14 | 461 |
| 5 | 1 | 15 | 1022 |
| 6 | 1 | 16 | 1968 |
| 7 | 3 | 17 | 3232 |
| 8 | 1 | 18 | 7228 |
| 9 | 15 | 19 | 13646 |

| Nivel de dificultate | Timp de execuție (ms) | Nivel de dificultate | Timp de execuție (ms) |
|---|---|---|---|
| 10 | 37 | 20 | 31962 |

Aşa cum observăm în figura 23, timpul de execuție urmează o curbă exponențială:

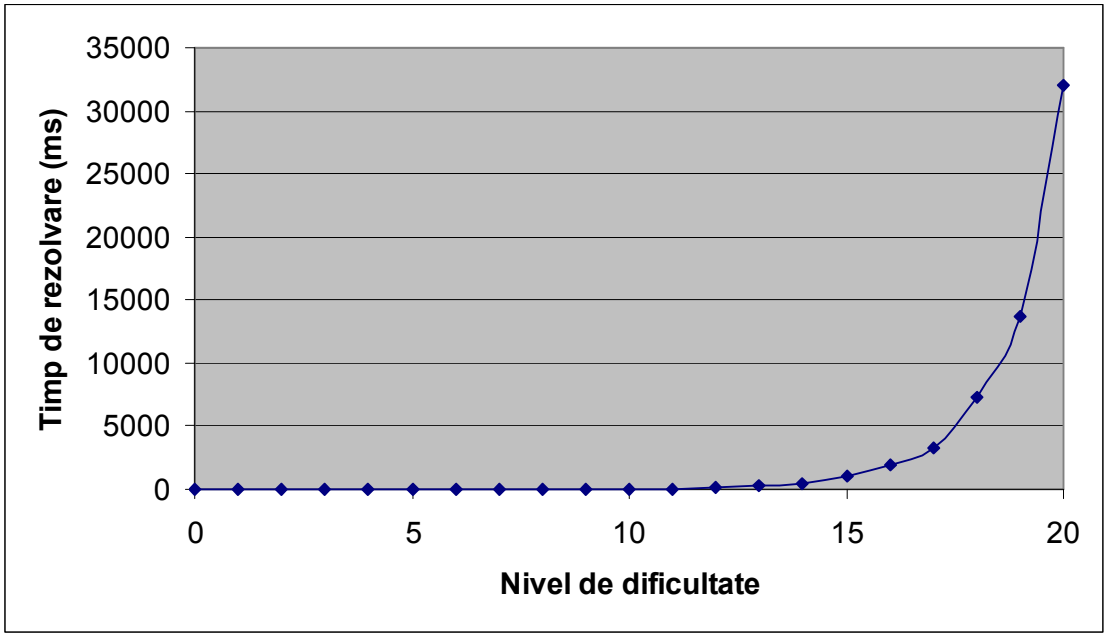


Figura 23. Curba timpului de execuție pentru diferite niveluri de dificultate

Pentru niveluri de dificultate peste 20, timpul de rezolvare al puzzle-ului devine prohibitiv, de aici aplicabilitatea practică redusă.

Pentru a obține o scală mai precisă pentru parametrul de dificultate al puzzle-ului, Juels şi Brainard [JUEL99] au propus împărțirea problemei în puzzle-uri mai mici de dificultate egală care ar urmau să fie rezolvate separat iar rezultatul total să fie obținut din combinarea rezultatelor parțiale. Aura, Nikkander şi Leiwo [TUOM00] susțin că aceeaşi granularitate poate fi obținută din combinarea sub-problemelor de dificultate diferită dar la un cost mai redus pentru server, fără a demonstra practic viabilitatea soluției.

## A1.5. Threshold puzzles

Client puzzles s-au dovedit eficace atât teoretic cât şi practic în condiții normale. Totuşi, dificultatea puzzle-ului este stabilită după o metrică ce ia în calcul doar angajamentul serverului şi nu ia în calcul puterea clientului care poate varia în limite largi. Aşa cum am văzut anterior, puzzle-urile sunt generate la momente precise, momente la care se ia în calcul angajamentul instantaneu al serverului. Această „măsură universală” nu este cea mai potrivită în toate scenariile.

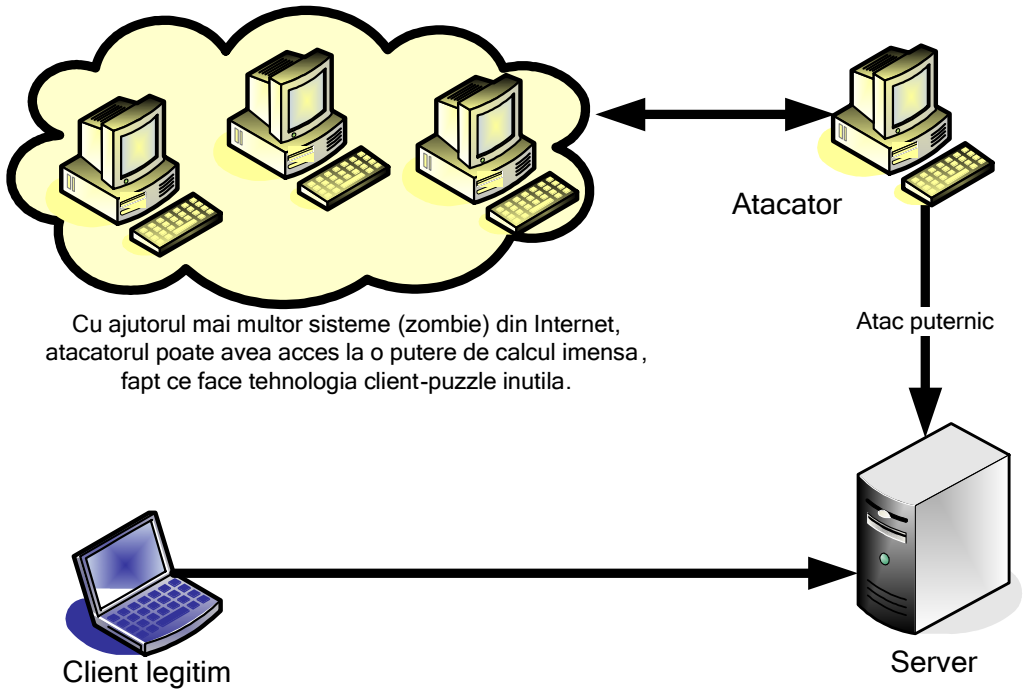


Figura 24. Schema unui atac puternic

Am remarcat că puzzle-urile sunt vulnerabile la o formă particulară de atac (numită de aici înainte **atac puternic**) datorită naturii paralele a puzzle-ului. Un atac puternic se defineşte ca un atac de tip *denial-of-service* pus la cale de un atacator cu acces la o putere de calcul imensă. În acest caz, atacatorul este capabil de a rezolva puzzle-urile într-un timp mult mai scurt decât un client legitim, după cum se poate vedea în figura 6.

Să presupunem că un server autentifică un număr de clienţi legitimi iar dificultatea iniţială a puzzle-ului este zero. Când un atac puternic este în desfăşurare, serverul are tendinţa de a creşte gradual nivelul de dificultate până la valori mari pentru a ţine pasul cu cantitatea importantă de procesare necesară pentru deservirea cererilor atacatorului. Deşi nivelul de dificultate poate fi crescut până la infinit, acest lucru înseamnă un atac DOS în sine îndreptat asupra clienţilor legitimi care nu ajung niciodată să rezolve un puzzle atât de dificil.

Deşi nu foarte probabil, un atac puternic este posibil. Dacă un atacator ar avea acces la N calculatoare (cu N suficient de mare ca să vorbim de putere de calcul imensă), atunci timpul necesar rezolvării puzzle-ului de dificultate k ar fi împărţit la N. Programul SETI [WWW4] şi efortul de a sparge algoritmul RSA [WWW5] sunt exemple reale de cum se pot pune la muncă sute de mii de sisteme pentru acelaşi scop comun. Puterea cumulată a reţelei *distributed.net* a depăşit echivalentul a 160000 de calculatoare PII la 266MHz, lucru ce arată clar că atacurile puternice sunt posibile.

Diferenţele aduse de tehnologia threshold puzzles faţă de client puzzles sunt:

- Limitarea superioară a nivelului de dificultate în aşa fel încât puzzle-ul rămâne în limite utile.
- Adăugarea unui timp minim de răspuns definiţiei puzzle-ului.

### A1.5.1. Limitarea superioară a nivelului de dificultate

Deşi design-ul curent al puzzle-ului aşa cum este el descris în [TUOM00] specifică o plajă a dificultăţii între 0 (nici un efort necesar) şi 128 sau 192 (imposibil, în funcţie de funcţia de dispersie folosită), o implementarea reală a mecanismului are mari şanse să aleagă o plajă mai rezonabilă, să spunem între 0 şi 25 datorită scalei exponenţiale care duce la margine îngustă a utilităţii. Niveluri mai ridicate ar putea să ducă la atacuri DOS îndreptate asupra clienţilor legitimi, lucru chiar mai grav decât atacul asupra serverului.

### A1.5.2. Stabilirea unui timp minim de răspuns

Ideea de bază constă în adăugarea unei amprente de timp la care serverul a generat valoarea sa aleatoare la lista «$N_S$, $N_C$, X, k». Când serverul primeşte soluţia, acesta poate calcula timpul exact de care a avut nevoie clientul pentru a rezolva puzzle-ul. Acest timp nu ar trebui să fie mai mic de o estimarea serverului bazată pe gradul de dificultate. Dacă este, atunci rezultă că serverul se află sub un atac puternic şi ar trebui să înceteze imediat comunicarea cu clientul în cauză. În medie, pentru rezolvarea unui puzzle de dificultate k este nevoie de $2^k - 1$ operaţii, deci formula de calcul al timpului estimativ este:

$$\mathbf{T_{estimat} = (2^k - 1) * T_{operaţie}}$$

- $T_{estimat}$ – timpul estimat pentru rezolvarea puzzle-ului
- k – nivelul de dificultate al serverului
- $T_{operaţie}$ – timpul minim de efectuare a unei operaţiuni criptografice (curent în plaja de 0.01 – 0.02 milisecunde, trebuie determinată experimental sau trebuie aplicată legea lui Moore la momentul implementării de fapt)

Timpul estimativ reprezintă pragul de acceptare pentru un client puzzle. Un client puzzle care încorporează modificările menţionate poartă numele de **threshold puzzle**.

## A1.6. Încărcarea serverului

Am văzut anterior că un puzzle constă din două cantităţi: o valoare aleatoare $N_S$ şi un nivel de dificultate k. Dacă prima cantitate nu pune probleme deosebite, riscurile folosirii unui generator de numere pseudo-aleatoare nefiind important, în cazul determinării nivelului de dificultate ne confruntăm cu o problemă. Nivelul de dificultate este o mărime strâns legată de încărcarea serverului. Cu cât încărcarea este mai mare, cu atât nivelul de dificultate este mai mare, intr-o creştere liniară. Gradul de încărcare este o noţiune complexă şi greu de estimat deoarece ne lovim de dificultatea găsirii unei metrici potrivite.

Dean şi Stubblefield [DEAN01] propun ca metrică numărul de operaţii RSA angajate. Contorul este incrementat când sistemul decide să nu trimită un puzzle sau când un client furnizează o soluţie corectă a unui puzzle generat anterior. Contorul se decrementează după ce operaţia RSA respectivă s-a încheiat sau dacă conexiunea s-a întrerupt înainte ca operaţia RSA să înceapă. Această metrică măsoară numărul de operaţii RSA care urmează să fie efectuate în perioada imediat următoare. Metrica are o creştere liniară, funcţia de calcul pentru parametrul k fiind:

$$\mathbf{k = N_{RSA} / Total_{RSA} * 100}$$

- $N_{RSA}$ – Numărul total de operaţii RSA angajate până în prezent

- $Total_{RSA}$ – Numărul maxim de operaţii RSA simultane pe care sistemul le poate executa fără degradarea performanţei
- k – nivelul de dificultate al puzzle-ului (între 0 şi 100)

Nivelul k calculat se aplică funcţiei de liniarizare prezentată anterior în lucrare pentru a obţine combinaţia optimă de puzzle-uri pentru client.

Pentru a acomoda şi sisteme distribuite (cum ar fi fermele de servere web) metrica ar trebui să ia în calcul încărcarea procesorului, capacitatea sistemului de I/O, schimbările de context, memoria liberă, numărul de întreruperi, etc. Elaborarea unei astfel de metrici este însă dincolo de scopul acestei lucrări.