Kerberos: Un serviciu de autentificare pentru sisteme de tip Open Network

Prioteasa Liviu Florentin, Rizescu Iulian Ștefan, Ţacu Ștefan Darius

Abstract

Acest proiect analizează tranziția de la mecanismele de autentificare seriale (locale) la cele distribuite, concentrându-se pe protocolul Kerberos. Într-o arhitectură de rețea deschisă, unde canalul de comunicație este inerent nesigur, transmiterea parolelor în clar reprezintă o vulnerabilitate critică. Lucrarea detaliază structura matematică a protocolului, bazată pe modelul Needham-Schroeder, și implementarea conceptului de Trusted Third Party (TTP). Sunt analizate fluxurile de schimb de mesaje (AS, TGS, CS), mecanismele de replicare Master-Slave și impactul asupra scalabilității în medii Enterprise, demonstrând eficiența utilizării criptării simetrice și a timestamp-urilor pentru securizarea sesiunilor.

1 Introducere

1.1 Context: Mediul "Open Network"

Apariția sistemelor distribuite, precum proiectul *Athena* de la MIT, a marcat o schimbare de paradigmă de la sistemele monolitice (Time-Sharing Systems) la arhitecturi de rețea deschise. Într-un astfel de mediu, stațiile de lucru (workstations) nu sunt fizic securizate, iar rețeaua care le conectează este vulnerabilă la interceptări.

Conform specificațiilor oficiale IETF [2], într-o rețea nesigură, un atacator poate monitoriza traficul ("sniffing") și poate intercepta pachetele de date. Astfel, orice protocol care se bazează pe transmiterea parolei în clar, sau chiar a hash-ului acesteia, este compromis din start. Mai mult, serverele nu pot avea încredere implicită în identitatea clienților doar pe baza adresei de rețea, care poate fi falsificată ("spoofing").

1.2 Definirea Problemei

Problema fundamentală pe care o adresăm este: *"Cum poate un utilizator să își dovedească identitatea unui serviciu de rețea fără a trimite parola prin canalul de comunicație nesigur și fără a stoca chei secrete pe fiecare server în parte?"*

Într-un sistem serial, nucleul sistemului de operare (Kernel) controlează atât terminalul utilizatorului, cât și resursa accesată, oferind o cale sigură (Trusted Path). Într-un sistem distribuit, această cale dispare. Avem nevoie de un mecanism care să garanteze identitatea părților comunicante fără a expune secretele acestora.

1.3 Soluția Propusă: Trusted Third Party (TTP)

Kerberos introduce conceptul de **Trusted Third Party** (TTP). În loc ca fiecare server să verifice parola utilizatorului (ceea ce ar implica stocarea parolelor pe sute de servere, mărind suprafața de atac), există o singură autoritate centrală în care toți au încredere: **KDC** (Key Distribution Center).

Modelul funcționează similar cu un sistem de legitimații: utilizatorul prezintă dovada identității o singură dată autorității centrale și primește un "tichet" criptat. Acest tichet este apoi prezentat diverselor servicii, care îl acceptă deoarece au încredere în semnătura emitentului (KDC)

[3].

1.4 Obiectivele Protocolului

Proiectarea Kerberos a urmărit patru obiective majore, esențiale pentru un sistem de autentificare robust:

1. **Securitate:** Un atacator care interceptează traficul nu trebuie să poată obține informații necesare pentru a impersona un utilizator.
2. **Fiabilitate (Reliability):** Sistemul de autentificare nu trebuie să devină un punct unic de eșec (Single Point of Failure) care să blocheze accesul la întreaga rețea. Acest lucru se realizează prin arhitectura distribuită Master-Slave.
3. **Transparență:** Utilizatorul trebuie să introducă parola o singură dată (Single Sign-On - SSO), restul procesului de obținere a tichetelor fiind invizibil.
4. **Scalabilitate:** Sistemul trebuie să suporte un număr mare de clienți și servere și să permită interconectarea între domenii administrative diferite (Cross-Realm Authentication) [2].

1.5 Concepte Cheie

Pentru a înțelege funcționarea algoritmului distribuit descris în Capitolul 3, definim următorii termeni:

- **Principals:** Entitățile care comunică (Utilizatori sau Servicii).
- **Tickets:** Structuri de date criptate care servesc drept permise de acces temporare.
- **Authenticators:** Informații generate pe loc (ce conțin timestamp-uri) pentru a dovedi că cel care prezintă tichetul este proprietarul legitim al acestuia.

2 Algoritmul Serial

În contextul autentificării, varianta serială a problemei presupune existența unui sistem monolitic în care utilizatorul și resursa accesată se află pe aceeași mașină fizică, sub controlul aceluiași nucleu (kernel). În acest scenariu, nu există o rețea de comunicații interpusă, iar mediul este considerat de încredere (Trusted Computing Base).

2.1 Descrierea Problemei

Problema fundamentală este verificarea identității unui *principal* (utilizator sau serviciu) înainte de a-i acorda acces la resursele sistemului. Autentificarea se bazează pe ceva ce utilizatorul *știe* (o parolă), care este comparată cu o valoare de referință stocată într-o bază de date securizată a sistemului.

2.2 Descrierea Algoritmului

Algoritmul serial de autentificare urmează un flux liniar de verificare a hash-urilor. Deoarece magistrala de date este internă și sigură, parola poate fi procesată direct, fără necesitatea unor tichete sau mecanisme de prevenire a atacurilor de tip *replay*, esențiale în varianta distribuită.

Pașii algoritmului sunt:

1. Utilizatorul furnizează identificatorul unic (*UserID*) și parola (*Password*).
2. Sistemul caută în tabela de *principals* (echivalentul `/etc/shadow` în UNIX) intrarea corespunzătoare identificatorului.
3. Dacă identificatorul nu există, accesul este refuzat imediat pentru a preveni scurgerea de informații despre existența conturilor.
4. Dacă utilizatorul există, se extrage cheia (sau hash-ul) stocată în baza de date (`key_bytes`).
5. Sistemul aplică funcția de derivare a cheii (KDF) asupra parolei introduse.

6. Se compară bit cu bit rezultatul obținut cu valoarea stocată.
7. Dacă valorile coincid, se inițializează o sesiune locală; altfel, se returnează o eroare de autentificare.

2.3 Pseudocod

Algoritmul poate fi formalizat după cum urmează:

Algorithm 1 Autentificare Serială (Locală)

```
procedure SerialAuth(UserID, InputPassword)
    record ← DB.lookup(primary_name = UserID)
    if record = NULL then return false                ▷ Utilizator inexistent
    end if
    DerivedKey ← KDF(InputPassword)
    if DerivedKey = record.key_bytes then
        CreateLocalSession(UserID) return true
    elsereturn false                                  ▷ Parolă incorectă
    end if
end procedure
```

3 Algoritmul Distribuit: Protocolul Kerberos

3.1 Fundamente Teoretice (Modelul Needham-Schroeder)

Protocolul Kerberos nu a apărut într-un vid, ci este o implementare și o rafinare a modelului de autentificare cu cheie simetrică propus de **Needham și Schroeder** [1].

Conceptul central preluat este cel de *Trusted Third Party* (TTP): un arbitru central în care toți clienții au încredere absolută. Sistemul funcționează pe premisa că, dacă KDC-ul confirmă identitatea unui utilizator, serviciile din rețea acceptă această judecată ca fiind corectă.

Inovația Kerberos (Timestamp vs. Nonce): O modificare fundamentală adusă modelului original este mecanismul de prevenire a atacurilor de tip *Replay* (unde un atacator interceptează un pachet valid și îl retrimite ulterior). În timp ce Needham-Schroeder se baza pe *nonce*-uri (numere aleatoare unice) și un schimb suplimentar de mesaje pentru a verifica prospețimea, Kerberos a introdus **Timestamp-urile** (marcaje temporale). Aceasta reduce traficul de rețea, dar introduce o dependență critică de sincronizarea ceasurilor.

3.2 Descriere Generală

Kerberos rezolvă problema autentificării într-o rețea nesigură folosind acest KDC. Pentru a face protocolul ușor de înțeles, îl putem privi ca pe un sistem de bilete: utilizatorul nu prezintă parola la fiecare serviciu, ci obține un ”permis universal” (TGT) pe care îl schimbă ulterior pentru ”bilete specifice” de acces.

3.3 Notații Utilizate

Pentru descrierea formală, vom folosi următoarele simboluri simplificate, conform lucrării originale [3]:

3.4 Fluxul Protocolului (Cele 3 Faze)

Protocolul este împărțit în trei etape logice.

Simbol	Semnificație
C, S	Clientul (utilizator) și Serverul (serviciul dorit)
AS, TGS	Componentele KDC: Authentication Server și Ticket-Granting Server
K_x	Cheia secretă a lui x (ex: hash-ul parolei)
$\{M\}_K$	Mesajul M criptat cu cheia K
TGT	*Ticket-Granting Ticket* (Permisul universal)
T_{Serv}	Tichetul de Serviciu (pentru acces final)
$Auth$	Autentificatorul (dovada identității pe loc, conține timestamp)

Table 1: Notații și Simboluri Kerberos

3.4.1 Faza 1: Autentificarea Inițială (Logarea)

Scop: Clientul vrea să demonstreze cine este, fără a trimite parola prin rețea.

În această fază, clientul inițiază protocolul pentru a obține credențialele necesare interacțiunii ulterioare. Figura de mai jos ilustrează schimbul de mesaje, unde axa timpului curge de sus în jos.

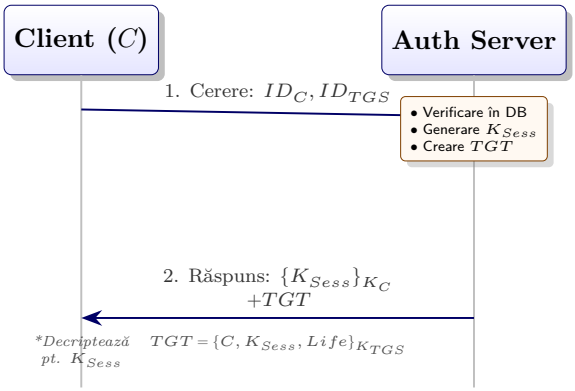

Figure 1: Diagrama de secvență pentru Autentificarea Inițială (AS Exchange)

După acest schimb, clientul deține cheia de sesiune decriptată și TGT-ul opac, fiind pregătit pentru Faza 2.

3.4.2 Faza 2: Obținerea Tichetului de Serviciu (TGS Exchange)

Scop: Clientul are TGT-ul (permisul) și vrea să acceseze un server specific (S), de exemplu un Server de Fișiere.

Utilizând cheia de sesiune obținută anterior, clientul generează un *Autentificator* pentru a dovedi că este posesorul legitim al TGT-ului.

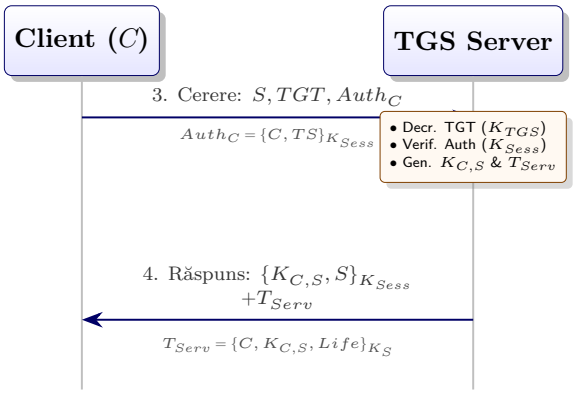

Figure 2: Diagrama de secvență pentru TGS Exchange

TGS-ul verifică timestamp-ul din autentificator pentru a preveni atacurile de tip *replay*. Dacă totul este valid, clientul primește noul tichet (T_{Serv}) și noua cheie de sesiune ($K_{C,S}$).

3.4.3 Faza 3: Accesarea Serviciului (Client-Server Exchange)

Scop: Clientul prezintă tichetul final serverului.

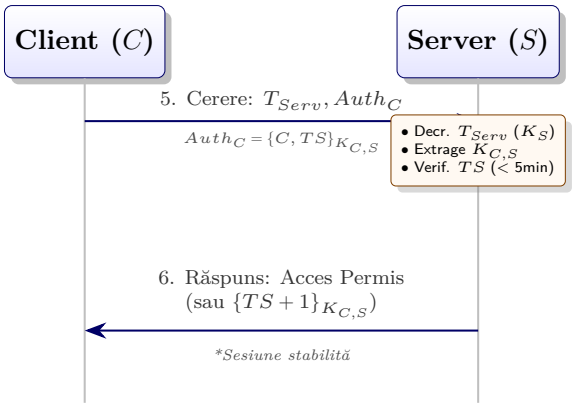

Figure 3: Diagrama Client-Server Exchange

3.5 De ce este acest sistem sigur?

1. **Parola nu circulă niciodată:** Ea este folosită doar local pe stația clientului pentru a decripta primul mesaj de la AS.

2. **Protecție la Replay:** Datorită Autentificatorului care conține *Timestamp*, un atacator care interceptează pachetele nu le poate refolosi mai târziu, deoarece serverul va respinge cererile cu ora veche.

3. **Minimizarea riscului:** Serverul de aplicație nu trebuie să știe parola utilizatorului, ci doar să poată decripta tichetul emis de KDC.

3.6 Replicarea și Administrarea Datelor

Pentru a asigura disponibilitatea și toleranța la defecte (fault tolerance), Kerberos nu rulează pe un singur nod, ci folosește o arhitectură distribuită de tip **Master-Slave** pentru baza de date de credențiale.

3.6.1 Arhitectura de Replicare (Master-Slave)

Baza de date Kerberos este centralizată logic, dar replicată fizic pentru disponibilitate. Arhitectura implică:

- **Master KDC:** Deține copia autoritativă (RW). Orice modificare (schimbare parolă, adăugare utilizator) se face aici.

- **Slave KDCs:** Dețin copii *Read-Only*. Acestea servesc cererile de autentificare (AS/TGS) pentru a reduce încărcarea pe Master.

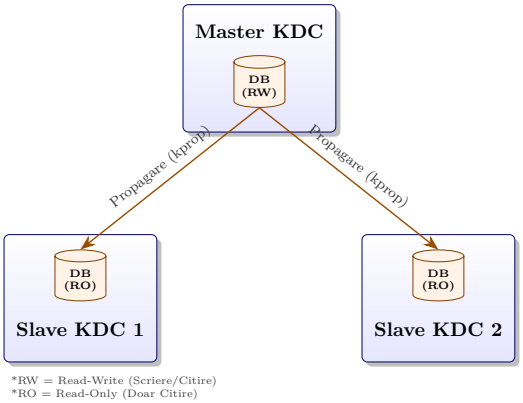

Figure 4: Arhitectura de Replicare Master-Slave

Mecanismul de Propagare: Conform specificației originale [3], consistența datelor este menținută prin propagare periodică (uzual la fiecare oră). Procesul de pe Master (`kprop`) trimite baza de date completă către procesele pereche (`kpropd`) de pe sclave. Pentru integritate, transferul este precedat de un **checksum criptat** cu cheia mașinii Master. Sclavul acceptă actualizarea doar dacă checksum-ul calculat local corespunde cu cel decriptat.

3.6.2 Protocolul de Administrare (KDBM)

Administrarea se face printr-un serviciu dedicat, **KDBM** (Kerberos Database Management), accesat prin utilitarele `kadmin` (pentru administratori) și `kpasswd` (pentru utilizatori simpli).

Acest serviciu are o particularitate de securitate critică:

> *"KDBM nu acceptă tichete emise prin TGS, ci impune ca*

utilizatorul să se autentifice direct prin serviciul AS."

Justificare de Securitate: Această constrângere impune utilizatorului să introducă parola **chiar acum**, demonstrând prezența fizică.

- Dacă s-ar permite folosirea unui TGT (care este stocat în memorie/cache), un atacator ar putea schimba parola unui utilizator care și-a lăsat stația de lucru nesupravegheată ("Passerby Attack").
- Forțând re-introducerea parolei pentru operațiuni sensibile, Kerberos mitigează riscul furtului de sesiune pentru acțiuni administrative.

4 Analiză Teoretică

Conform cerințelor de proiectare, evaluăm protocolul din perspectiva corectitudinii, performanței și scalabilității.

4.1 Analiză Corectitudine

Corectitudinea protocolului Kerberos se bazează pe două premise esențiale care diferențiază acest sistem distribuit de varianta serială:

1. **Sincronizarea Ceasurilor (Time Skew):** Deoarece autentificatorii ($Auth_C$) se bazează pe timestamp-uri pentru a preveni atacurile de tip *replay*, este critic ca ceasurile clientului și serverului să fie sincronizate (de obicei prin protocolul NTP). O deviație mai mare de 5 minute (configurație standard) duce la respingerea tichetelor valid emise, cauzând o eroare de disponibilitate (False Negative).
2. **Integritatea TTP (Trusted Third Party):** Securitatea întregului sistem depinde de faptul că KDC-ul nu este compromis. Deoarece KDC-ul stochează cheile secrete ale tuturor principalilor, compromiterea acestuia permite unui atacator să genereze orice tichet (atac de tip "Golden Ticket"), permițând impersonarea oricărui utilizator către orice serviciu.

4.2 Analiză Complexitate Timp / Mesaj

Pentru o sesiune completă de autentificare și acces la un serviciu ("Cold Start"), complexitatea mesajelor este constantă, totalizând 6 pachete de rețea:

- **AS Exchange:** 2 mesaje (Cerere + Răspuns).
- **TGS Exchange:** 2 mesaje (Cerere + Răspuns).
- **Client-Server:** 2 mesaje (Cerere + Confirmare).

Optimizare (Cache): Ulterior, pentru accesarea aceluiași serviciu, costul scade la **2 mesaje**, deoarece clientul refolosește tichetul de serviciu (T_{Serv}) până la expirarea acestuia (uzual 8 ore). Pentru accesarea unui *serviciu nou*, costul este de **4 mesaje** (TGS + CS), deoarece se refolosește TGT-ul.

Din punct de vedere al complexității comunicației, protocolul este extrem de eficient, având un număr constant de pași ($O(1)$), indiferent de numărul total de utilizatori din sistem. Din punct de vedere computațional, încărcarea asupra procesorului este redusă, deoarece protocolul utilizează exclusiv criptare simetrică (DES/AES), care este semnificativ mai rapidă decât operațiile matematice complexe necesare în criptarea asimetrică (RSA/ECC).

4.3 Comportamentul la Variația Numărului de Noduri (Replicarea Master-Slave)

Scalarea sistemului prin adăugarea de noduri suplimentare (Slave KDCs) prezintă următoarele caracteristici tehnice:

- **Avantaje (Read Throughput):** Adăugarea de servere Slave crește liniar capacitatea de procesare a cererilor de autentificare (AS/TGS). Deoarece majoritatea traficului în Kerberos este de tip *Read-Only* (verificarea credențialelor și emiterea de tichete), sistemul scalează eficient pentru un număr mare de utilizatori simultani.
- **Dezavantaje (Write Bottleneck & Propagation Lag):** Indiferent de numărul de noduri Slave adăugate, arhitectura impune o limitare critică:
    1. **Scrieri Exclusive pe Master:** Orice modificare a bazei de date (schimbarea parolei, adăugarea unui utilizator) trebuie procesată **exclusiv** de KDC-ul Master. Astfel, Master-ul rămâne un *Single Point of Failure* (SPOF) pentru operațiunile de scriere. Dacă Master-ul cade, utilizatorii se pot autentifica (prin Slave), dar nu își pot schimba parolele.
    2. **Latența de Propagare:** Există un decalaj (*propagation lag*) între momentul scrierii pe Master și actualizarea tuturor Slave-urilor. În acest interval, un utilizator ar putea să se autentifice pe un Slave folosind vechea parolă, deși a schimbat-o pe Master.

4.4 Exemplificare Topologii

Pe lângă arhitectura standard (Client-Server-KDC), protocolul suportă topologii complexe pentru medii enterprise.

4.4.1 Topologie Cross-Realm

Aceasta permite unui client dintr-un domeniu (Realm A) să acceseze servicii dintr-un alt domeniu (Realm B), cu condiția să existe o relație de încredere (cheie partajată) între cele două servere KDC.

În figura 5, fluxul este modificat astfel:

1. Clientul cere KDC-ului local un tichet pentru KDC-ul străin.
2. KDC-ul local emite un ”Remote TGT” criptat cu cheia inter-realm.
3. Clientul prezintă acest TGT direct KDC-ului străin și primește tichetul de serviciu (T_{Serv}).
4. Clientul prezintă T_{Serv} serverului final pentru acces.

4.4.2 Topologie de Delegare (Proxy)

O inovație majoră descrisă în lucrarea originală Athena [3] este mecanismul de *Proxy*. Aceasta rezolvă problema în care un serviciu intermediar trebuie să acceseze o resursă în numele utilizatorului.

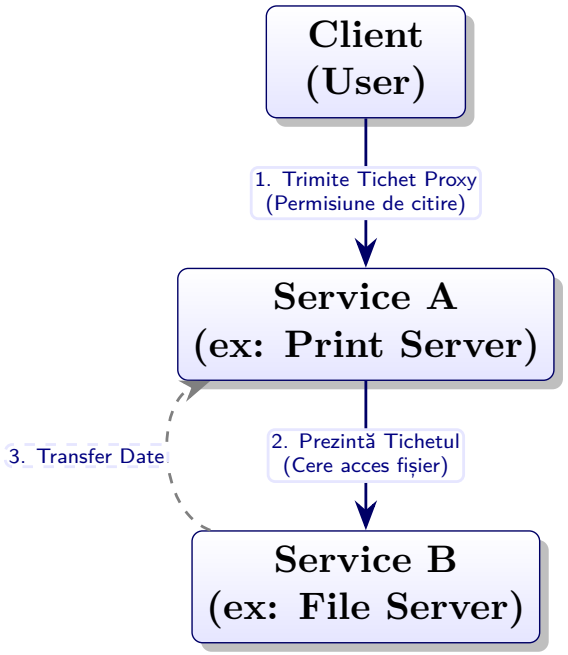

Figure 6: Topologie Proxy: Delegarea drepturilor între servicii

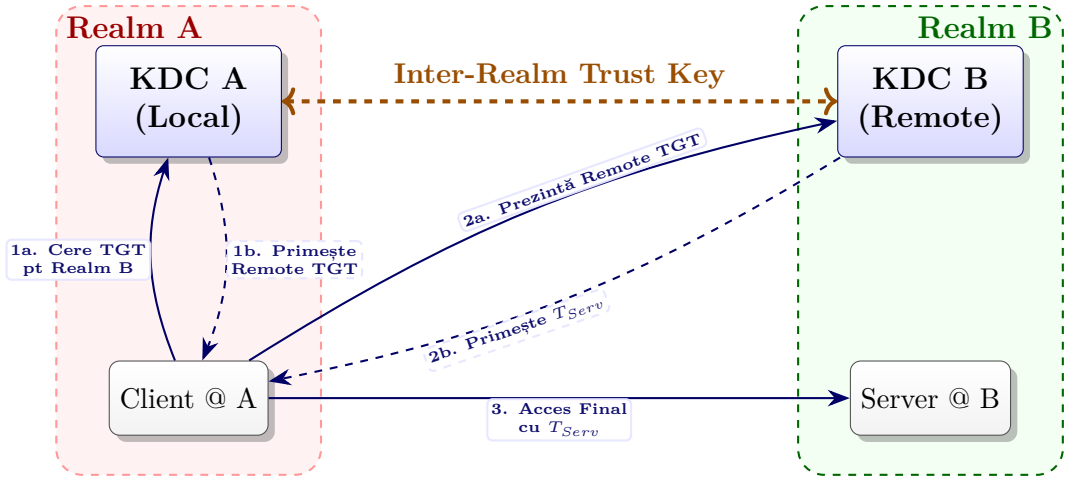

Figure 5: Topologie Cross-Realm: Autentificare între domenii diferite

Această arhitectură simplifică managementul accesului în sisteme distribuite complexe, eliminând necesitatea autentificării multiple.

References

[1] Roger M. Needham and Michael D. Schroeder. "Using encryption for authentication in large networks of computers". In: *Communications of the ACM* 21.12 (Dec. 1978), pp. 993–999.

[2] Dr. Clifford Neuman, Sam Hartman, Kenneth Raeburn, and Taylor Yu. *The Kerberos Network Authentication Service (V5)*. RFC 4120. July 2005. DOI: 10.17487/RFC4120. URL: https://www.rfc-editor.org/info/rfc4120.

[3] Jennifer G Steiner, B Clifford Neuman, and Jeffrey I Schiller. "Kerberos: An authentication service for open network systems." In: *Usenix Winter*. 1988, pp. 191–202.