UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Role Based Access Control Systems

DIPLOMOVÁ PRÁCA

informatika

Autor: Ľubomír Teťák
Vedúci diplomovej práce: RNDr. Martin Stanek, PhD. Bratislava 2006

Zadanie práce

Cieľom práce má byť prehľad systémov riadenia práv založených na rolách RBAC, implementácia RBAC modulu a jeho použitie ako modul riadenia práv v FTP serveri.

Abstrakt

RBAC (Role Based Access Control) je jedným z možných spôsobov riadenia prístupu v operačných systémoch, databázových systémoch, internetových aplikáciách a pod. Práca predstavuje NIST RBAC štandard a jeho komponenty. Porovnáva niekoľko systémov riadenia prístupu pre OS Linux a RDBMS s komponentmi NIST RBAC štandardu. Práca popisuje základné atribúty RBAC systémov. Súčasťou práce je implementácia základného a hierarchického komponentu NIST RBAC štandardu ako modulu riadenia práv v FTP serveri.

Predhovor

S príchodom počítačových sietí s viacpoužívateľskými operačnými systémami a programami vznikol problém riadenia prístupu v týchto systémoch. Jedným z možných riešení problému riadenia práv v počítačových systémoch je *riadenie prístupu založené na rolách* (RBAC), ktorému sa venujeme v našej práci. Porovnávame rôzne implementácie RBAC a popisujeme ich atribúty. Za odrazový mostík sme si zvolili NIST RBAC štandard. Porovnávame s ním niektoré implementácie a implementujeme jeho dva komponenty. Dúfame, že výsledná aplikácia bude inšpiráciou pre ďalšie RBAC implementácie.

Prehlásenie

Prehlasujem, že na práci som pracoval samostatne, s použitím uvedenej literatúry.

Obsah

1. Úvod do systémov riadenia prístupu

Rast počtu počítačových systémov a zvyšovanie množstva ich používateľov za posledné desaťročia si vyžiadal vývoj od prvých samostatných počítačov po dnešné celosvetové siete ako je Internet. Určenie kto má možnosť pristupovať v sieti k danému prostriedku sa stalo veľmi dôležité pre uchovanie súkromia a dôvernosti informácií. Preto vznikli systémy riadenia prístupu. Používateľ systému môže získať prístup k dátam alebo na vykonanie operácie. Najbežnejším príkladom programu, ktorý riadi prístup je operačný systém (OS). V minulosti ale aj dnes na domácich počítačoch riadi OS prístup lokálne na každom počítači zvlášť. Takýto spôsob je najjednoduchší na implementáciu, ale vyžaduje si veľké náklady na administráciu už pri niekoľkých počítačoch prepojených do siete. Preto sa moderné systémy (OS, databázové systémy, internetové aplikácie, ...) vytvárajú s myšlienkou centralizovaného riadenia prístupu.

Technológie riadenia prístupu sa časom zdokonaľovali, pričom sa vyčlenili dva hlavné typy: *voliteľné riadenie prístupu (DAC – Discretionary Access Control)* a *povinné riadenie prístupu (MAC – Mandatory Access Control)*.

DAC dáva možnosť jednotlivým používateľom prideľovať a odoberať prístupové práva [1]. V tejto schéme môžu byť používatelia taktiež vlastníkmi objektov. Systémy založené na DAC dovoľujú používateľom povoliť alebo zakázať prístup k ľubovoľným objektom, ktoré vlastnia. Problém môže nastať, ak používateľ nie je skutočným majiteľom informácie, ktorej vlastníkom je v DAC systéme (príkladom môže byť lekár, ktorý má právo čítať lekárske záznamy pacienta, ale nie je ich skutočným majiteľom a teda nemá právo ich ďalej poskytnúť tretej strane – toto právo má len pacient). Často skutočným majiteľom informácií (objektov), ako aj programov, ktoré tieto informácie spracúvajú, je samotná organizácia. Preto je v niektorých prípadoch výhodnejší použiť iný typ riadenia prístupu napr. MAC, kde sú práva prístupu riadené organizáciou.

MAC je spôsob riadenia prístupu k objektom založený na stupni dôvernosti informácie v nich obsiahnutej (stupeň dôvernosti určuje organizácia a nie vlastník informácie) a autorizácii používateľa pristupovať k informáciám označeným určitým stupňom dôvernosti [1]. To znamená, že organizácia určí kto môže pristupovať ku informáciám určitého stupňa dôvernosti. Riadenie prístupu je povinné v zmysle, že používateľ nemá právo meniť stupeň dôvernosti informácie a tým ju poskytnúť používateľom s menšími právami.

Ani jeden z uvedených typov riadenia práv však nie je optimálny pre organizácie so zložitou štruktúrou spracúvajúce dôverné informácie. Vhodnejším typom riadenia prístupu sa javí riadenie prístupu založené na rolách (RBAC), čo je vlastne rozšírený typ MAC.

Za cieľ práce sme si položili vytvoriť prehľad existujúcich systémov využívajúcich RBAC, nájsť spoločné atribúty v babylone RBAC systémov, implementovať RBAC modul a následne ho použiť v FTP serveri.

Všeobecnému popisu RBAC sa venuje nasledujúca kapitola. V kapitolách 3 a 4 sa bližšie pozrieme na existujúce implementácie niekoľkých komerčných i voľne dostupných systémov využívajúcich RBAC. Napriek tomu, že všetky sú založené na rovnakom princípe (rozdelenie používateľov do skupín podľa ich roly a priradenie práv skupinám), značne sa odlišujú. Nejednotnosť RBAC systémov sa snažila riešiť americká organizácia NIST vypracovaním štandardu pre RBAC [4]. Je podrobne popísaný v 5. kapitole. Aby sme zistili skutočné možnosti NIST RBAC štandardu, vytvorili sme implementáciu hierarchického RBAC modulu. Tento modul sme následne použili v jednoduchom FTP serveri. Snažili sme sa o novátorské riešenie niektorých aspektov implementácie (napr. diagram hierarchie rolí). Najzaujímavejšie z nich sú popísané v 6. kapitole. Z práce vidieť, že rôznorodých implementácii RBAC systémov je mnoho, v 7. kapitole sme sa preto pokúsili definovať atribúty, ktoré spájajú tieto systémy a sú použiteľné pre hodnotenie možností RBAC systémov (použili sme ich pri hodnotení našej implementácie). Vďaka špecifikácii týchto atribútov je možné napríklad lepšie popísať požiadavky pri výbere existujúcej implementácie RBAC poprípade špecifikovať požadované vlastnosti novej implementácie. Súčasťou práce je priložené CD. Jeho obsah je popísaný v prílohe 1. V prílohe 2 sa nachádza niekoľko rád pri inštalácii FTP servera z CD.

2. Systémy riadenia prístupu založené na rolách (RBAC)

Napriek tomu, že sa používajú a vyvíjajú rôzne systémy riadenia prístupu, práve systémy riadenia prístupu založené na rolách (RBAC) vyzerajú byť sľubnou metódou na riadenie toho, ktoré informácie môžu používatelia spracovať a aké operácie môžu vykonať.

2.1. Čo je to RBAC?

Prístup definujeme ako možnosť vykonávať činnosť s počítačovým prostriedkom (napr. vykonať, zmeniť, zobraziť súbor) [1].

Riadenie prístupu je spôsob, akým je povolený alebo zakázaný prístup. Predpisuje nielen to, kto môže mať prístup ku systémovému prostriedku, ale aj typ povoleného prístupu (napr. pri súborovom systéme právo čítať zo súboru, zapisovať do súboru, zmazať súbor, atď.).

Pri RBAC je riadenie prístupu založené na rolách, ktoré presne mapujú funkcie jednotlivých používateľov v organizácii. Používatelia majú priradené roly podľa svojej funkcie, napr. účtovník, lekár, profesor, pracovník personálneho oddelenia alebo manažér. Proces definovania rolí by mal byť založený na priamej analýze fungovania organizácie a mal by zahŕňať poznatky od širokého spektra používateľov v organizácii. Správna definícia rolí je kľúčová k efektívnej prevádzke RBAC systému, keďže sa predpokladá, že roly sa nebudú často meniť.

Prístupové práva sú priraďované jednotlivým rolám a používanie prostriedkov (objektov) je vyhradené len pre používateľov oprávnených zastávať priradenú rolu. Napríklad v rámci fakturačného systému rola predajcu môže zahŕňať operácie vytvárania a administrácie vlastných faktúr, zatiaľ čo rola kontrolóra môže byť limitovaná na zobrazenie informácií o všetkých faktúrach bez možnosti modifikácie dát.

2.2. Používatelia

Používateľ podľa NIST štandardu je definovaný ako ľudská bytosť. Aj keď koncept používateľa môže byť rozšírený tak, aby zahŕňal zariadenia v sieti, procesy OS alebo inteligentných autonómnych agentov, pre jednoduchosť je definícia zúžená len na osoby.

Používatelia v RBAC systémoch získavajú príslušnosť k rolám, čím získavajú povolenie na vykonávanie operácií na objektoch systému priradených danej role. Pri administrácii RBAC systémov je jednoduché vytvoriť používateľa a priradiť mu existujúcu

rolu alebo podobne odobrať mu existujúcu rolu a priradiť inú (napr. pri zmene funkcie zamestnanca).

Obvyklou bezpečnostnou požiadavkou v počítačových systémoch je, že používateľovi nesmie byť povolené viac, ako je nevyhnutné na vykonanie operácií potrebných pre vykonávanie práce v jeho funkcii. Toto pravidlo sa volá *princíp najmenšej množiny privilégií.* Postup definovania práv jednotlivých používateľov v organizácii si vyžaduje zistenie všetkých požadovaných funkcií, ktoré bude daný používateľ pre svoju prácu potrebovať. V organizáciách s voľnejšou štruktúrou to môže byť problém a používateľ tak dostane viac práv, ako je nevyhnutné. Ak používateľ zastáva viac postov, získa práva všetkých postov, čím môže získať možnosť vykonať operácie, na ktoré by nemal mať právo (napr. agent v poisťovni môže byť súčasne jej klient. Nemal by však mať možnosť sám si vystaviť zmluvu). Tieto problémy sa dajú riešiť niekoľkými spôsobmi:

- Dohodou mimo systému, napríklad nepovoliť používateľovi zastávať obe roly.
    - V našom príklade by mohla mať poisťovňa v každej zmluve klauzulu, podľa ktorej nie je možné, aby si agent sám vystavil zmluvu. Bolo by však nutné nejakým spôsobom kontrolovať jej dodržiavanie, čo môže byť neefektívne.
- Statickým vylúčením práv, čiže definovaním množín vzájomne sa vylučujúcich rol
    - Pri priradení používateľa do roly klienta poisťovne skontrolovať, či nie je agentom tejto poisťovne. Ak áno, neumožniť mu stať sa jej klientom. Toto riešenie je efektívnejšie ako predchádzajúce, ale môže zbytočne obmedzovať používateľov.
- Dynamickým vylúčením práv, čiže definovaním množín rolí, ktoré nemôže používateľ súčasne zastávať počas trvania jedného prihlásenia.
    - Pri vytváraní zmluvy skontrolovať, či ju agent nevydáva sám sebe.

Statické a dynamické vylúčení práv je podrobne popísané v kapitole 5 venovanej NIST štandardu, ktorý tieto funkcie podporuje.

2.3. Roly a hierarchie rolí

Rola je podľa NIST štandardu funkcia, ktorá v prostredí organizácie prideľuje používateľom práva a zodpovednosť. Nejde teda len o syntaktické zoskupenie používateľov.

Pri administrácii práv jednotlivých rolí sa často stane, že používatelia patriaci do rôznych rolí môžu vykonávať rovnaké operácie a teda viaceré roly budú mať rovnaké spoločné množiny práv. Ak by roly nemali medzi sebou definované žiadne vzťahy, museli by sme každej z nich priradiť tieto práva samostatne, čo nie je časovo efektívne a prispieva to

k neprehľadnosti systému. Obvykle sa preto zavádza relácia čiastočného usporiadania, ktorá vytvára hierarchiu rolí odzrkadľujúcu štruktúru organizácie. Napríklad práva roly zamestnanca fakulty získajú aj roly profesor a pracovník na ekonomickom oddelení. Alebo naopak, rola pracovníka na personálnom oddelení získa práva zamestnanca fakulty aj zamestnanca ekonomického oddelenia, ktoré spadá pod jeho kompetenciu v štruktúre organizácie.

Výhodou hierarchie rolí je, že v reálnom prostredí organizácií je zväčša dobre definovaná hierarchia funkcií pracovníkov alebo ich oddelení. Pre administrátora systému je teda jednoduché a prirodzené vytvoriť túto hierarchiu aj v RBAC systéme a určiť každej role prístupové práva. Pri priradení roly používateľovi si tak môžeme byť istí, že získal minimálnu množinu oprávnení, ktorú potrebuje pre vykonávanie svojej práce.

Zmeny v RBAC systéme sa vykonajú rýchlejšie a spoľahlivejšie, ak roly odrážajú skutočnú štruktúru organizácie. Napríklad zmena roly doktoranda zo študenta na zamestnanca katedry je pri dobrom návrhu štruktúry RBAC systému jednoduchá.

2.4. Operácie a práva

Podľa NIST štandardu je *operácia* definovaná ako spustiteľný obraz programu, ktorý po vyvolaní vykoná pre používateľa určitú funkciu. V širšom chápaní to teda môže byť každá funkcia programu.

Ako už z pomerne všeobecnej definície operácie vidno, RBAC systém môže obsahovať veľmi rôznorodé operácie. Najčastejšie sa však snažíme dodržiavať určité štandardné množiny operácií v danom obore (napr. v súborovom systéme sú to práva na prácu so súbormi a adresármi ako vytvoriť, čítať, zapísať, atď.).

Keď chceme priradiť role právo vykonávať určitú operáciu, väčšinou musíme špecifikovať aj objekt, na ktorom má byť táto operácia vykonaná. Je rozdiel, keď má používateľ právo zápisu v svojom domovskom adresári a v systémovom adresári. Preto je RBAC systém zväčša navrhnutý tak, že definuje dvojice objekt – operácia, ktoré nazývame *právo*. Rolám sa potom priradia tieto práva.

2.5. Prihlásenia

Prihlásenie (session) je množina operácií vykonávaných v časovom intervale od vstupu používateľa do systému po jeho odhlásenie.

RBAC podľa NIST štandardu musí umožňovať používateľovi prihlásiť sa do systému a postupne si aktivovať roly (to znamená, že nemusia byť všetky aktívne v momente vstupu

do systému). Postupné aktivovanie rolí je užitočné hlavne pri použití vylúčenia práv (kapitoly 5.3 a 5.4). Jeden používateľ musí mať možnosť prihlásiť sa do systému súčasne viac krát, pričom každé prihlásenie má vlastnú množinu aktívnych rolí.

2.6. Výhody RBAC

Administrácia systémov riadenia prístupu môže byť nákladná a náchylná na chyby. Ferraiolo, Barkley a Kuhn [2] štatisticky zistili, že najčastejšími operáciami pri administrácii systémov riadenia prístupu je presun používateľa z jednej pracovnej funkcie do inej (častejšie ako vytváranie novej funkcie alebo zmena práv funkcie). V RBAC systémoch funkcii používateľa zodpovedá rola v systéme. Presun používateľa do inej funkcie je v systéme vykonaný priradením používateľovi inej roly. V štandardných systémoch funkcii zodpovedá množina práv priradených konkrétnemu používateľovi a preto presun používateľa si vyžaduje priradenie novej množiny práv, čo je zväčša náročnejšia operácia. Najväčšia zložitosť administrácie v RBAC systémoch je teda sústredená do vytvárania hierarchie rolí a systému práv na začiatku používania systému. Následné zmeny sú už jednoduché. RBAC systémy sú teda lepšie aplikovateľné v organizáciách s relatívne stálou štruktúrou.

Ďalšou výhodou priradenia práv role a nie používateľovi je to, že jednu funkciu zväčša zastávajú viaceré osoby a na výkon svojej práce potrebujú väčšinou viac ako jedno právo. Je preto rýchlejšie priradiť viacerým osobám tú istú rolu ako každej osobe priraďovať zvlášť množinu práv, ktoré potrebuje na výkon svojej práce. Približný odhad zníženia nákladov na administráciu oproti bežným systémom je možné vypočítať programom na internetovej stránke NIST-u [15].

Stručný prehľad rozdielov medzi RBAC a štandardnými systémami riadenia prístupu zobrazuje nasledujúca tabuľka.

RBAC	**Štandardné systémy riadenia prístupu**
• Rola je pomenovaná množina používateľov, práv a ďalších rolí. Nemusí obsahovať žiadnych používateľov [17].	• Skupina je pomenovaná množina používateľov, práv a ďalších skupín, ktorá spravidla obsahuje aspoň dvoch používateľov [17].
• Roly mapujú štruktúru organizácie (Ján Novák – účtovník, mzdové oddelenie)	• Skupiny nie sú priamo zviazané so štruktúrou organizácie (Ján Novák – používateľ s obmedzenými právami)

• Prístupové práva sú priradené rolám (účtovník, mzdové oddelenie, atď.). • Používateľom sú priradené roly • Očakávame častejšie zmeny funkcií používateľov ako štruktúry organizácie.	• Prístupové práva sú priradené používateľom (Ján Novák, atď.). • Používatelia sú rozdelení do skupín

Tab. 1: Porovnanie RBAC a štandardných systémov riadenia prístupu

Výhody RBAC v porovnaní so štandardnými systémami riadenia prístupu sú:

- Rýchla a flexibilná administrácia
  - Novému používateľovi vieme s veľkou pravdepodobnosťou priradiť už existujúcu rolu v systéme (nie je nutné opätovné priraďovanie všetkých potrebných práv používateľovi).
  - Presun používateľa do inej funkcie si vyžaduje iba presun do inej roly (študent – doktorand).
  - Vďaka hierarchii rolí nie je nutné vytvárať množstvo rolí s prelínajúcimi sa právami.
- Prehľadnosť
  - Pracuje sa s pojmami prirodzenými pre človeka (lekár, zdravotná sestra, agent, manažér, atď.).
  - Možnosť jednoduchej grafickej administrácie hierarchie rolí.
- Nové možnosti ohraničení
  - Statické a dynamické vylúčenie práv.

3. Vybrané implementácie systémov riadenia práv

Systém riadenia práv OS Linux by sme mohli zaradiť do systémov typu DAC (pozri kapitolu 1) s tým rozdielom, že v systéme existuje jeden používateľ s neobmedzenými právami *root*. Väčšine programov postačujú práva bežného používateľa systému. Sú však programy, ktoré vyžadujú vyššie práva a často musia byť spustené s právami roota (napr. niektoré sieťové programy – *nmap, tcpdump*, atď). Musíme teda týmto programom povoliť plný prístup k systému, čo porušuje princíp minimálnych práv. Nie vždy máme možnosť overiť si činnosť takýchto programov (napr. nahliadnutím do zdrojového kódu). Je teda výhodnejšie rozdeliť práva jedného *root* používateľa na viacerých napríklad pomocou niekoľkých systémových rolí. Práve túto činnosť si za cieľ zobrali nasledujúce projekty. Budeme sa sústrediť hlavne na ich moduly zabezpečujúce riadenie prístupu.

Informácie som čerpal najmä z domovských stránok projektov a použitej literatúry.

3.1. RSBAC

Domovská stránka projektu: [9]

Použitá literatúra: [6]

Implementácia RSBAC (*Rule Set Based Access Control*) sa skladá z hlavného modulu, ktorý sa nachádza priamo v jadre Linuxového systému a modulov, ktoré implementujú jednotlivé bezpečnostné modely (MAC, ACL, RC, atď.) a modulov zabezpečujúcich iné funkcie (AUTH, UM, atď.). Vďaka takejto modulárnej architektúre je RSBAC značne univerzálny (je možné relatívne jednoducho vytvoriť nový modul zatiaľ nepodporovaného bezpečnostného modelu). Jeho veľká viazanosť na jadro Linuxu ho však robí ťažko prenositeľným na OS s iným jadrom.

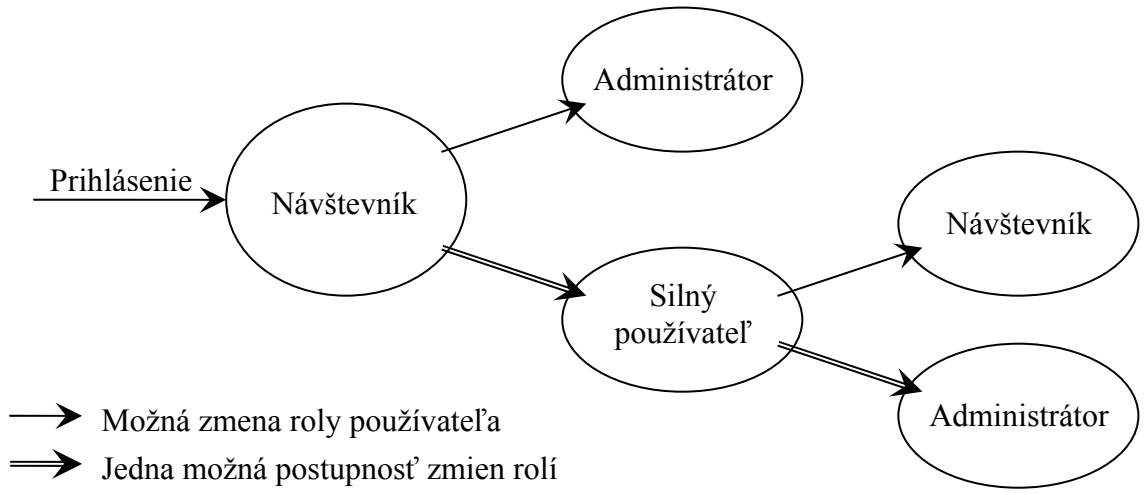
Moduly RSBAC:

- Autentifikácia používateľa (*AUTH*)
- Správa používateľov (*UM*) – nahradzuje pôvodný systém riadenia práv používateľov a skupín Linuxu niektorým (jedným alebo viacerými) z množiny modelov podporovanými RSBAC
- Model kompatibilných rolí (*RC*) – systém založený na prideľovaní práv rolám. Je v mnohom podobný s RBAC. Je podrobne popísaný nižšie.

- Povinné riadenie prístupu (*MAC*) – pri spustení priradí procesu bezpečnostnú úroveň používateľa. Proces tak získa možnosť prístupu na čítanie k procesom, súborom alebo zariadeniam na rovnakej alebo nižšej bezpečnostnej úrovni a možnosť zápisu k objektom na rovnakej úrovni. Práva sú pridelené systémom a nie vlastníkom objektu.

Model kompatibilných rolí (RC)

Tento model obsahuje entity rolí a ich typov. Počet rolí je prakticky neobmedzený. Typ roly môže byť súbor, adresár, rúra (FIFO objekt), symbolický odkaz, atď. Každá rola obsahuje množinu kompatibilných rolí. Používateľ má po prihlásení sa priradenú a aktívnu jedinú predvolenú rolu. Ak táto rola obsahuje množinu kompatibilných rolí, môže si aktivovať jednu z nich. Aktívnu však môže mať súčasne vždy len jednu rolu. Po zmene roly nemusí byť zmena späť možná. Je tým zakázaný nežiaduci tok informácií, keď používateľ z roly s nižšími právami prejde do roly s vyššími právami, získa informácie a prejde naspäť do predchádzajúcej roly, kde informácie odovzdá používateľom s nižšími právami.



Obr. 1: Zmena roly používateľa v modeli kompatibilných rolí (RC).

Nech má používateľ pri prihlásení priradenú predvolenú aktívnu rolu *návštevník* (tak, ako je to zobrazené na Obr. 1). Rola *návštevník* má množinu kompatibilných rolí *administrátor* a *silný používateľ*. Môžeme teda aktivovať napríklad rolu *silný používateľ*. Táto rola má kompatibilné roly *návštevník* a *administrátor*. Aktivujeme rolu *administrátor*. Rola *administrátor* ma prázdnu množinu kompatibilných rolí, preto v tomto prihlásení už nemá používateľ možnosť zmeniť rolu.

V niektorých prípadoch môžeme požadovať, aby na vykonanie úlohy administrátora boli potrební viacerí administrátori (napr. reštartovanie servera, vytvorenie

administrátorského konta, atď.). *Vylúčenie práv administrátorov* je spôsob, ktorým rozdelíme postup administrácie systému niekoľkým používateľom (administrátorom), ktorí musia pre vykonanie úloh administrácie spolupracovať.

V RC systémoch je možné definovať životnosť priradenia roly, po vypršaní nadefinovaného časového limitu budú používateľovi pridelené kompatibilné roly odstránené.

Porovnanie RC s NIST RBAC

RC podobne ako RBAC môže obsahovať rôzne subjekty ako napríklad procesy (v RBAC sú to používatelia vykonávajúci proces), ktoré majú priradenú rolu [10]. V RC môže mať proces súčasne iba jednu aktívnu rolu (čím je odstránený problém vzájomného vylúčenia práv, ktoré rieši RBAC pomocou statického vylúčenia rolí – SSD).

Množinu autorizovaných rolí používateľa z RBAC nahrádza množina kompatibilných rolí v RC, ktoré môže používateľ dosiahnuť z predvolenej roly.

RBAC neumožňuje definovať časové limity platnosti roly, roly programov a vylúčenie administrátorských práv.

RC model neumožňuje definovať hierarchiu rolí v zmysle RBAC modelu.

Pomocou RC modelu je možné dosiahnuť takmer všetky funkcie, ktoré umožňuje RBAC a aj niektoré iné (životnosť priradenia roly, vylúčenie práv administrátorov). Je ťažké povedať, ktorý z nich je intuitívnejší. RBAC je však rozšírenejší, preto jeho princípy budú používateľom pravdepodobne známejšie ako princípy RC modelu.

Model RC je vhodné použiť vtedy, keď je objektom možné priradiť skupinu a jej typ, ak je potrebné silné vylúčenie práv alebo ak je potrebné riadiť práva procesov a nie len používateľov.

3.2. Medusa DS9

Domovská stránka projektu: [7]

Použitá literatúra: [11]

Projekt Medusa DS9 má počiatky na Slovensku v roku 1997. Jeho tvorcovia sa vtedy zaoberali otázkou, ako univerzálne zabezpečiť OS Linux pred stále častejšími útokmi.

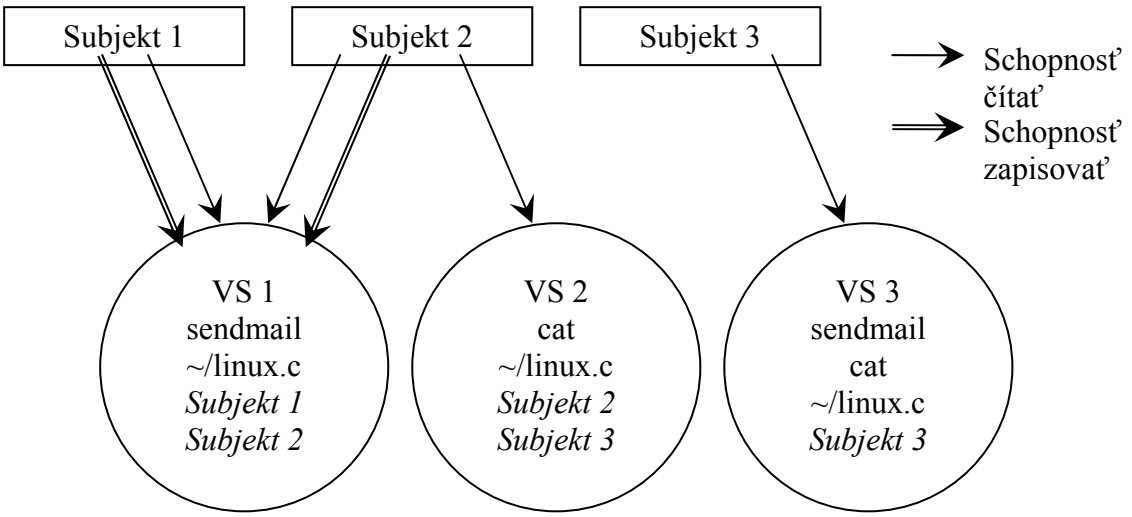
Hlavné nedostatky Linuxu videli v príliš veľkých oprávneniach root-a, nemožnosti definovať podrobne práva pre niektoré objekty v systéme (najmä sieťové služby), slabých možnostiach delegovania práv objektmi iným objektom, slabých možnostiach definovania relatívnych bezpečnostných pravidiel a prístupových práv.

Záplaty na konkrétne diery v systéme sa im zdali nepostačujúce, preto postupovali inak. Definovali objekty, subjekty a operácie (prístupy) systému. Výsledkom je *popis systému* (definícia objektov, subjektov a typov prístupov) a *bezpečnostný model* (ktorý popisuje, ako je možné prideľovať alebo odoberať prístupové práva).

V zatiaľ poslednej stabilnej implementácii Medusa DS9 pre Linux 2.4.x je možné pracovať s objektmi: systémové volania, súbory, procesy a IPC objekty (rúra), subjektmi: proces a vlákno (*thread*) a operáciami: čítanie (*read*), zápis (*write*) a videnie (*see* – zistiť existenciu objektu).

Medusa využíva ako bezpečnostný model tzv. *VS model* (*virtual spaces* model). Objekty a subjekty sú rozdelené do konečného počtu domén (nazývaných virtuálne svety – oddelenie nezávislých častí systému). Každý objekt môže byť zaradený do ľubovoľného počtu virtuálnych svetov a každý svet môže obsahovať niekoľko objektov. Každý subjekt má priradenú množinu schopností (jednu pre každý typ prístupového práva). Schopnosť určuje množinu virtuálnych svetov, ku objektom ktorých má subjekt právo pristupovať. Ak objekt a subjekt zdieľajú aspoň jeden spoločný virtuálny svet, subjektu je priradené právo na prístup ku objektu.

Ukážeme si tento model na zjednodušenom príklade. Chceme vykonať sériu príkazov *cat ~/linux.c | sendmail lubo@tetak.sk*. Na Obr. 2 sú subjekty znázornené obdĺžnikmi, virtuálne svety kružnicami, objekty sú texty v kružniciach a schopnosť je znázornená šípkou od subjektu ku virtuálnemu svetu. Subjekty 2 a 3 majú schopnosť čítať priradenú ku VS obsahujúcim všetky potrebné objekty (*cat* a *~/linux.c*), ale iba subjekt 2 má priradenú potrebnú schopnosť zapisovať do VS obsahujúceho príkaz *sendmail*. Subjekt 3 teda nemôže vykonať danú sériu príkazov. Subjekt 1 nemá právo vykonať príkaz *cat* a tiež nemôže vykonať celú sériu príkazov.



Obr. 2: Model virtuálnych svetov.

Model virtuálnych svetov je kompatibilný s RBAC modelom, len pomenúva entity inak a pozerá sa na systém z iného uhla pohľadu (pomocou Tab. 2 je možné syntakticky previesť systém medzi VS modelom a RBAC modelom).

RBAC model	**VS model**
množina objektov	virtuálny svet
objekt	objekt
rola	subjekt
právo	schopnosť

Tab. 2: Porovnanie pomenovania entít RBAC a VS modelu.

Vývoj projektu napreduje pomaly (v súčasnosti neexistuje stabilná verzia pre jadro Linuxu 2.6.x), napriek tomu sa systém používa. Hlavným cieľom projektu Medusa DS9 bolo vytvoriť univerzálny systém, ktorý by mohol využívať ľubovoľný bezpečnostný model (alebo dokonca spájať rôzne bezpečnostné modely napríklad pomocou logických operátorov $\wedge$ a $\vee$).

3.3. SELinux

Domovská stránka projektu: [19]

Použitá literatúra: [6]

Na rozdiel od predošlých dvoch projektov začal SELinux (*Security Enchanced Linux*) ako projekt vyvíjaný organizáciou National Security Agency (NSA). V roku 2000 uvoľnila NSA zdrojové kódy SELinuxu verejnosti, ale naďalej sa zapája do jeho vývoja. SELinux je rovnako ako Medusa a RSBAC skupina úprav (záplat) jadra Linuxu ale okrem zmeny jadra obsahuje aj upravené niektoré utility systému (*ssh, cron, ls, ps*, atď.).

Cieľom SELinuxu je umožniť riadiť práva OS Linux pomocou MAC (na rozdiel od súčasného typu riadenia DAC).

SELinux pozostáva z komponentov TE a RBAC. TE (T*ype Enforcement*) komponent definuje množinu domén a typov. Proces má priradenú doménu, objekt má priradený typ. Následne umožňuje špecifikovať relácie doména-typ a doména-doména, ktoré určujú prístup domény k objektu alebo k inej doméne. Podobne ako RSBAC umožňuje TE špecifikovať povolené prechody procesu medzi doménami ako aj predvolenú doménu pre daný typ.

RBAC komponent priradí každému procesu rolu a množinu domén, ktoré môžu byť nadobudnuté danou rolou. Po prihlásení do systému je používateľovi podobne ako v RSBAC pridelená predvolená doména. Rozdiel medzi RBAC v SELinuxe a NIST RBAC je ten, že SELinux roly používa iba pre pridelenie domény používateľom a nie priamo prideľovanie práv ako NIST RBAC.

4. Porovnanie vybraných RBAC DBMS

V predošlej kapitole sme popísali spôsoby implementácie RBAC v troch nadstavbách operačného systému Linux. V tejto kapitole porovnáme možnosti RBAC piatich systémoch na správu dát (DBMS). Keďže ide o zložité systémy, väčšinu informácií z prehľadu prvých troch sme prevzali z práce Ramaswamyho a Sandhua [14] (táto práca je voľne dostupná na internetovej stránke, preto je popis týchto systémov stručný). Našim cieľom bolo porovnať iba vlastnosti systémov a nie príkazy, ktorými sa vykonávajú. Príkazy sú preto uvedené iba v zátvorke kvôli ľahšej orientácii. Popis posledných dvoch z pätice DBMS systémov (dnes pomerne rozšíreného Microsoft SQL Server-a a PostgreSQL) sme vypracovali podľa ich príručiek [8] a [13].

Môžeme si klásť otázku, prečo potrebujeme v databázovom serveri také silné nástroje na riadenie prístupu. Tvorcovia DBMS systémov sa domnievajú, že práve to je najlepší spôsob zabezpečenia systému. Bezpečnostný model je implementovaný priamo pri zdroji dát a bezpečnostná politika je definovaná centrálne (čo je opačný prístup ako jej uloženie zvlášť v každej aplikácii využívajúcej rovnakú databázu). Ak je nutné bezpečnostnú politiku upraviť, stačí ju zmeniť na jednom mieste a bude platiť pre všetky aplikácie využívajúce danú databázu nezávisle na spôsobe pripojenia k databáze.

V závere kapitoly je prehľadná tabuľka porovnávajúca tieto systémy.

4.1. INFORMIX Online Dynamic Server Version 7.2

Informix umožňuje priradiť rolu jednému, skupine alebo všetkým používateľom (PUBLIC). Jeden používateľ môže mať priradených viac rolí, ale aktívnu môže mať najviac jednu z nich (SET ROLE). Základný model NIST RBAC štandardu vyžaduje mať možnosť postupnej aktivácie viacerých rolí, preto Informix nespĺňa základný model. Každý používateľ s priradenou rolou (GRANT OPTION) môže priradiť túto rolu ďalším používateľom alebo rolám. Môže tiež zrušiť túto rolu (DROP ROLE). Vytvorením relácie medzi rolami môže vytvoriť hierarchiu rolí. Informix neumožňuje definovať statické ani dynamické vylúčenia práv.

V Informixe sa práva delia do troch skupín: databázové, tabuľkové a vykonávacie.

- Databázové práva sa vzťahujú na pripojenie k DB (CONNECT), správu objektov DB (RESOURCE), správu práv a vlastníctva objektov DB (DBA) a správu pamäťového priestoru vyhradeného databáze.

- Tabuľkové práva zahŕňajú operácie vzťahujúce sa tabuľkám DB (SELECT, UPDATE, INSERT, DELETE, ALTER, INDEX, REFERENCES).
- Vykonávacie právo je právo vykonať (EXECUTE) uloženú procedúru v DB (*SP – stored procedure*).

4.2. Sybase Adaptive Server release 11.5

DBMS od Sybase obsahuje okrem používateľských rolí aj preddefinované roly pre administráciu systému nazvané systémové roly. Sú to sa-role (administrátor systému), sso-role (administrátor systémovej bezpečnosti), oper-role (operátor).

Priradenie používateľov k roliam je rovnaké ako v Informixe s tým rozdielom, že ho smie vykonávať iba používateľ zastávajúci rolu sso-role alebo vlastník objektu. Jeden používateľ si môže v rámci prihlásenia postupne aktivovať niekoľko rolí (SET ROLE). Môže si taktiež vytvoriť preddefinovaný zoznam rolí, ktoré sa mu aktivujú pri prihlásení (SP_MODIFYLOGIN).

Možnosť vytvoriť hierarchiu rolí je rozšírená o možnosť, ktorú nemá žiaden z vybraných DBMS a tou je *statické* a *dynamické vylúčenie práv*.

- Dve roly sa *staticky* vylučujú, ak používateľ nemôže mať priradené obe súčasne.
- Dve roly sa *dynamicky* vylučujú, ak používateľ nemôže mať aktívne obe súčasne.

Vylúčenie práv v Sybase Adaptive Server neponúka všetky možnosti, ktoré ponúka NIST RBAC štandard. Tento DBMS však má možnosť definovať maximálny počet rolí, ktoré môže mať používateľ naraz aktívne (NIST RBAC toto obmedzenie neumožňuje).

Sybase rozdeľuje práva na práva pre prístup k objektom a práva pre vytváranie objektov.

4.3. Oracle Enterprise Server Version 8.0

Priradenie používateľov a práv je rovnaké ako v Informixe (namiesto GRANT OPTION sa používa príkaz ADMIN OPTION).

Aktivácia rolí (SET ROLE) oproti Sybase umožňuje spojiť niekoľko volaní do jedného, výsledný efekt však bude rovnaký. To isté platí o vytvorení zoznamu prednastavených rolí pri prihlásení (ALTER USER, DEFAULT ROLE).

Rovnako ako v prípade Informixu Oracle nepodporuje statické ani dynamické vylúčenie práv a takisto ani obmedzenie počtu používateľov roly.

Oracle rozdeľuje práva na systémové a objektové. Systémové riadia vykonávanie všetkých príkazov určitého typu bez ohľadu na to, na ktorom objekte budú vykonané (SELECT ANY TABLE), objektové riadia vykonávanie príkazov na konkrétnom objekte databázy (tabuľke, pohľade, sekvencii, SP alebo funkcii).

Oracle 9i prináša ďalšie rozšírenie rolí nazývané *bezpečné aplikačné roly*. Bezpečné aplikačné roly sú vytvárané špecifikovaním mena balíčku, do ktorého prislúcha rola.

4.4. Microsoft SQL Server 2000

Na trhu už je nástupca verzie SQL Servera 2000 od Microsoftu s označením 2005. V oblasti rolí však neprináša (okrem zväčšenia počtu preddefinovaných systémových rolí) žiadnu novú funkcionalitu. Pozrieme sa preto na momentálne rozšírenejšiu verziu SQL Server 2000.

SQL Server od Microsoftu obsahuje až štyri skupiny rolí. Sú to serverové, databázové, aplikačné a štandardné (používateľské) roly. SQL Server môže tiež využívať roly a používateľov hosťovského OS.

- Serverové roly sú pevne definované a platia v celom systéme. Ich zoznam získame pomocou SP *sp_helpsrvrole*. Majú pevne priradené práva nad všetkými databázami v kontexte celého systému.
- Databázové roly sú tiež pevne definované, ale platia len v kontexte databázy, ktorá obsahuje danú rolu. Ich zoznam získame pomocou SP *sp_helpdbfixedrole***.**
- Aplikačné roly nemajú priradených žiadnych používateľov. Majú priradenú aplikáciu, po vytvorení spojenia z tejto aplikácie a aktivovaní roly zadaním povinného hesla získa používateľ aplikácie príslušnosť k aplikačnej role (a tým aj všetky jej práva). Je tak napríklad možné zakázať pristupovať k databáze priamo z Enterprise Managera bez použitia vlastnej aplikačnej vrstvy. Používané systémové SP: *sp_addapprole, sp_setapprole, sp_dropapprole.*
- Štandardné roly sú podobné ako roly v ostatných troch porovnávaných systémoch. Používateľovi môžeme priradiť viac rolí a rola môže mať viac používateľov. Po prihlásení má používateľ aktivované všetky priradené roly.

4.5. PostgreSQL server

Systém rolí a používateľov PostgreSQL servera je kvôli multiplatformovej architektúre nezávislý od operačného systému. Roly sú vytvárané v kontexte servera alebo skupiny previazaných serverov (CREATE ROLE). Nie je možné vytvárať roly v kontexte jedinej databázy.

Rola môže mať niekoľko atribútov:

- login – iba rola, ktorá má tento atribút môže byť použitá pre prihlásenie do systému
- superuser – rola s týmto atribútom má všetky práva v systéme
- database creation – tento atribút povolí role vytvoriť novú databázu v systéme
- role creation – umožní role spravovať iné roly (okrem rolí s atribútom superuser)
- password – rola má priradené heslo

Okrem týchto atribútov môžeme role nastaviť preddefinované systémové parametre (napríklad vypnúť zaznamenávanie činnosti v log súboroch, kontrolu indexov pri štarte a mnohé ďalšie). Tieto parametre sa nastavia pri prihlásení používateľa a platia len pre jeho prihlásenie.

Zoznam rolí a ich atribútov je uložený v systémovej tabuľke *pg_roles*.

Každý objekt v systéme má priradeného vlastníka, čo je zvyčajne rola, ktorá objekt vytvorila. K objektu má prístup iba rola s atribútom superuser, vlastník objektu a roly, ktorým bol prístup povolený (GRANT). Výnimkou je objekt s atribútom PUBLIC, ku ktorému má povolený prístup každá rola.

Používateľovi v systéme môže byť priradená rola a roly môžu vytvárať hierarchiu. Používateľ si môže aktivovať rolu dvoma spôsobmi. Ak je člen roly, príkazom SET ROLE ju aktivuje (deaktivuje sa mu však rola, ktorú mal pri prihlásení). Objekty, ktoré vytvorí s takto aktivovanou rolou budú mať vlastníka túto rolu. Druhý spôsob umožňuje role nastaviť atribút INHERITED. Takáto rola bude jej členom (priamym aj zdedeným v hierarchii) automaticky aktivovaná pri prihlásení.

4.6. Zhrnutie

Funkcia	**Informix**	**Sybase**	**Oracle**	**MS SQL**	**PostgreSQL**
Možnosť používateľa zastávajúceho rolu priradiť túto rolu iným používateľom	Áno	Nie	Áno	*Áno*	*Áno*
Možnosť vlastníka objektu priradiť práva na objekt iným rolám	*Áno*	*Áno*	*Áno*	*Áno*	*Áno*
Možnosť aktivácie viacerých rolí	Nie	Áno	Áno	*Áno*	*Áno*
Možnosť špecifikácie predvolenej množiny aktívnych rolí pre prihlásenie	Nie	Áno	Áno	*Nie*	*Áno*
Hierarchia rolí	Áno	Áno	Áno	*Áno*	*Áno*
Statické vylúčenie práv	*Nie*[1]	Áno	Nie	*Nie*	*Nie*
Dynamické vylúčenie práv	Nie	Áno	Nie	*Nie*	*Nie*
Možnosť nastaviť max/min počet používateľov roly	Nie	Nie	Nie	*Nie*	*Nie*
Možnosť role priradiť systémové práva DBMS	Nie	Áno	Áno	*Áno*	*Áno*
Možnosť role priradiť objektové práva DBMS	Áno	Áno	Áno	*Áno*	*Áno*

[1] Práca [14] uvádza, že Informix podporuje statické vylúčenie práv, keďže umožňuje mať súčasne aktívnu iba jednu rolu. Nemyslíme si však, že to spĺňa definíciu vylúčenia práv podľa NIST štandardu, ktorý v tejto práci považujeme za smerodajný.

Funkcia	Informix	Sybase	Oracle	MS SQL	PostgreSQL
Spĺňa predpoklady pre základný model NIST RBAC	*Nie*[2]	*Áno*	*Áno*	*Áno*	*Áno*
Spĺňa predpoklady pre hierarchický model NIST RBAC	*Áno*[3]	*Áno*	*Áno*	*Áno*	*Áno*
Kompatibilita s NIST RBAC SSD modelom	*Nie*	*Áno*	*Nie*	*Nie*	*Nie*
Kompatibilita s NIST RBAC DSD modelom	*Nie*	*Áno*	*Nie*	*Nie*	*Nie*

Tab. 3: Porovnanie možností DBMS systémov (Prevzaté z práce [14] okrem riadkov a stĺpcov vyznačených kurzívou).

V Tab. 3 je vidieť, že DBMS systémy podporujú RBAC v rôznom rozsahu. Väčšina spĺňa požiadavky kladené NIST RBAC štandardom pre základný a hierarchický model (Sybase DMBS podporuje všetky štyri modely). Cez DBMS získavame silný nástroj na riadenie prístupu pre mnoho systémov (intranetové systémy, webové aplikácie, ap.).

V praxi sa však roly často nevyužívajú a riadenie prístupu sa prenecháva na vyššie vrstvy aplikácie. Nemalú zásluhu na tom určite má aj cena, ktorú musí klient zaplatiť za licenciu pre každého používateľa nachádzajúceho sa v DBMS systéme (samozrejme to platí iba pre komerčné DBMS). Najčastejšie databázu využíva iba jediná aplikácia a preto centralizované riadenie prístupu nebýva vždy potrebné. Rozšírené riadenie prístupu pomocou RBAC môžeme nájsť na serveroch ponúkajúcich webhosting internetových aplikácií. Mnoho používateľov s často citlivými údajmi (napr. elektronické obchody) zdieľajú ten istý databázový server a preto požadujú zabezpečenie databázy už na dátovej vrstve. Niektoré riešenia ako napríklad aplikačné roly MS SQL servera sa tiež dajú ťažko nahradiť iným spôsobom.

Z iných rozšírených DMBS systémov spomenieme MySQL, ktorý však RBAC nepodporuje (a podľa autorov v blízkej dobe ani nebude podporovať).

[2] Základný model NIST RBAC štandardu vyžaduje mať možnosť súčasnej aktivácie niekoľkých rolí.

[3] Hierarchický model NIST štandardu predpokladá prítomnosť základného modelu, ktorý Informix nespĺňa kvôli chýbajúcej možnosti súčasnej aktivácie viacerých rolí.

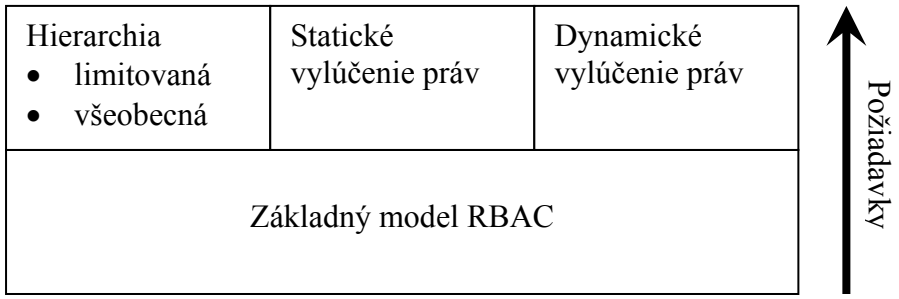
5. NIST RBAC štandard

Táto kapitola vysvetľuje základné koncepty RBAC štandardu NIST-u. Návrh NIST-u bol prijatý ako Americký národný štandard inštitúciou American National Standards Institute, International Committee for Information Technology Standards (ANSI INCITS 359-2004) v roku 2004.

Aplikácie využívajú často iba niektoré funkcie RBAC a ostatné pre ne nie sú potrebné. NIST RBAC štandard preto definuje štyri komponenty RBAC modelu:

- základný model
- hierarchický model
- statické vylúčenie práv
- dynamické vylúčenie práv

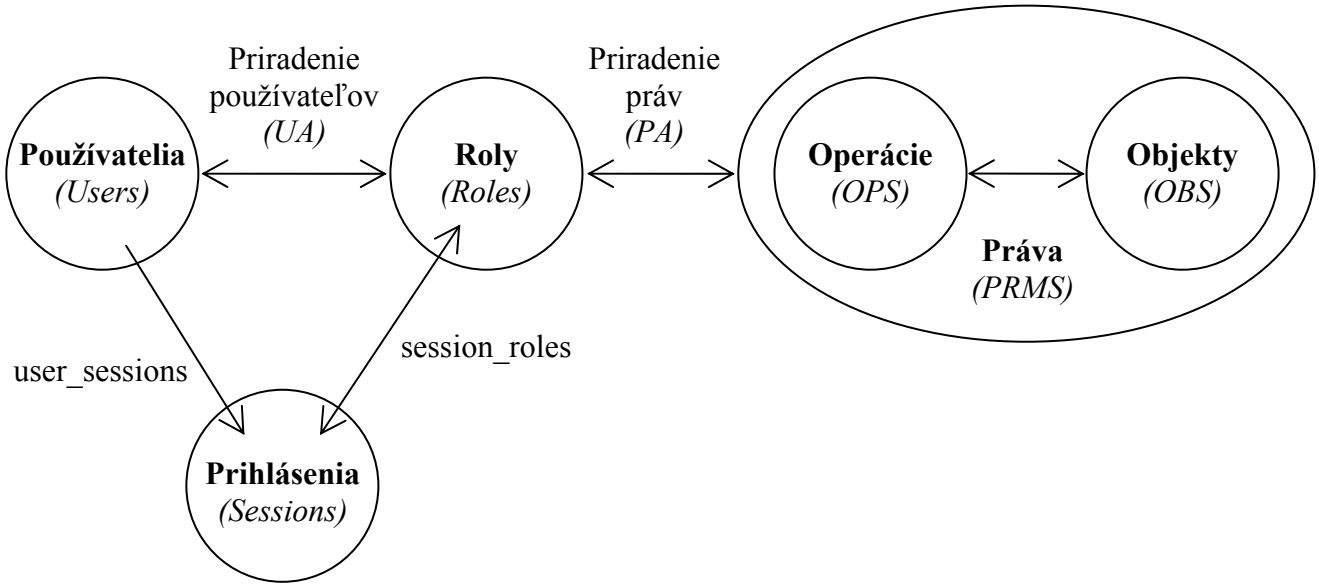
Základný model zahrňuje funkcie nevyhnutné pre každú RBAC aplikáciu. Ostatné tri komponenty sú už od seba nezávislé a využívajú len funkcie zo základného modelu.



Obr. 3: Komponenty RBAC modelu.

NIST RBAC je aj vďaka tejto modulárnej architektúre vhodný ako referenčný štandard, s ktorým je možné porovnávať rôzne RBAC systémy. Stačí si zvoliť množinu funkcií, ktoré nás zaujímajú. Pri implementácii RBAC systému je tiež výhodné, že netreba implementovať celý štandard, ale iba využívané komponenty (napr. ak systém nepožaduje statické vylúčenie práv, nemusíme ho implementovať).

5.1. Základný model RBAC



Obr. 4: Základný model RBAC.

Základný model RBAC obsahuje tieto množiny entít (v RBAC systémoch štandardne označované uvedenými skratkami):

U – používatelia (Users)

R – roly (Roles)

P alebo PRMS – prístupové práva (Permissions)

OPS – operácie

OBS – objekty

Medzi týmito množinami definuje tieto relácie:

UA – priradenie používateľov (User Assignment) je relácia typu n:n medzi používateľmi a rolami.

PA – priradenie práv (Permission Assignment) je relácia typu n:n medzi prístupovými právami a rolami

S – prihlásenia (Sessions) je relácia typu 1:n medzi používateľmi a rolami

Z najdôležitejších množín vytvorených reláciami nad základnými množinami spomenieme množinu $assigned\_users(r:ROLES) \rightarrow 2^{USERS}$, ktorá predstavuje množinu všetkých používateľov priradených k role *r*. Formálne

$$assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}.$$

Základný model RBAC obsahuje jadro funkcií, ktoré využíva každá RBAC aplikácia. Ako je zobrazené na Obr. 4, používatelia sú priradení k rolám, práva sú priradené k rolám a používatelia získavajú práva príslušnosťou k rolám. Používateľ môže mať priradených niekoľko rolí a rola môže obsahovať viacerých používateľov. Podobne rola môže obsahovať viac práv a viaceré roly môžu obsahovať to isté právo. Jeden používateľ môže naraz využívať viac prihlásení, ale viac používateľov nemôže zdieľať jedno prihlásenie. Implementácia základného modelu musí obsahovať funkcie, ktoré vedia pracovať s týmito reláciami.

Podmienkou základného modelu je umožniť používateľovi súčasne využívať práva viacerých rolí. To si vyžaduje možnosť aktivácie niekoľkých rolí súčasne v rámci jedného alebo viacerých prihlásení. Pri prihlásení sa používateľov aktivuje predvolená rola alebo všetky priradené roly.

Pri bližšom pohľade na základný model RBAC si uvedomíme, že sa podobá na typ riadenia prístupu MAC (popísaný v kapitole 1). Podobný spôsob riadenia prístupu (odhliadnuc od vlastníctva objektov systému) je využívaný v mnohých súčasných operačných systémoch, napr. Linux a Microsoft Windows (iba serverové edície umožňujú vytvoriť hierarchiu skupín). Nie každý typ riadenia prístupu ale spĺňa zároveň všetky vyššie popísané požiadavky pre základný RBAC model. Funkcie týchto OS však netvoria podmnožinu ani nadmnožinu funkcií RBAC. Napríklad Linux štandardne neumožňuje priradiť viacerým skupinám právo čítať zo súboru. Oba OS však napríklad umožňujú priradiť práva aj používateľovi, nie len skupine. Rozdielom medzi RBAC a inými systémami riadenia prístupu sa čiastočne venuje kapitola 2.6.

Štandard definuje RBAC pomocou množín funkcií, ktoré musí každá aplikácia obsahovať (administratívne, systémové a prehľadové) a množiny funkcií, ktoré sú nepovinne voliteľné (rozšírené prehľadové funkcie). Jednotlivé modely RBAC systému dedia tieto funkcie v poradí: základný model, hierarchický model, statické vylúčenie práv, dynamické vylúčenie práv. Nie je teda nutné v každom modely implementovať nanovo všetky funkcie, niektoré môžeme použiť napríklad zo základného modelu. Každá funkcia má formálne definované vstupné a výstupné premenné a vstupné a výstupné podmienky. Vďaka formalizmu je štandard jednoznačný a súčasne umožňuje voľnosť použitých algoritmov pri implementácii.

Administratívne funkcie slúžia na vytvorenie a administráciu základných množín. Základné množiny sú množiny používateľov, rolí, objektov, operácií a práv (predpokladá sa, že množiny objektov a operácií sú preddefinované systémom využívajúcim RBAC).

Systémové funkcie riadia prihlásenia a umožňujú nastaviť a zistiť práva k objektom systému.

Prehľadové a rozšírené prehľadové funkcie umožňujú monitorovať systém, neumožňujú meniť entity systému (rozšírené prehľadové funkcie nie sú povinné pri implementácii).

Prehľad základných entít RBAC systému (používatelia, roly, objekty a operácie) tak isto ako rozšírené vlastnosti týchto entít (heslo používateľa, popis roly, atď.) získa aplikácia priamo z dátovej vrstvy z príslušných tabuliek v databáze. Napríklad používateľa vytvoríme zavolaním funkcie *AddUser*, ktorej návratová hodnota je objekt nového záznamu tabuľky *Users* v databáze. V tomto zázname následne vyplníme ostatné vlastnosti (skutočné meno, heslo a popis). Zoznam všetkých rolí získame z tabuľky *Roles* v databáze. Prehľadové funkcie zabezpečujú iba zložitejšie operácie (obdoba pohľadov – *view* v databázových systémoch) a neobsahujú tieto triviálne zoznamy

a) Administratívne funkcie

AddUser, DeleteUser, AddRole, DeleteRole – pridá alebo odstráni zo systému používateľa alebo rolu

AssignUser, DeassignUser – vytvorí alebo zruší priradenie roly a používateľa

GrantPermission, RevokePermission – vytvorí alebo zruší priradenie roly a práva

b) Systémové funkcie

CreateSession, DeleteSession – pridá alebo odstráni zo systému prihlásenie s množinou aktívnych rolí

AddActiveRole, DropActiveRole – aktivuje alebo deaktivuje rolu pre dané prihlásenie

CheckAccess – funkcia vráti booleovskú hodnotu určujúcu, či prihlásený používateľ má právo vykonávať danú operáciu na danom objekte

c) Prehľadové funkcie

AssignedUsers – vráti množinu používateľov priradených k danej role

AssignedRoles – vráti množinu rolí priradených k danému používateľovi

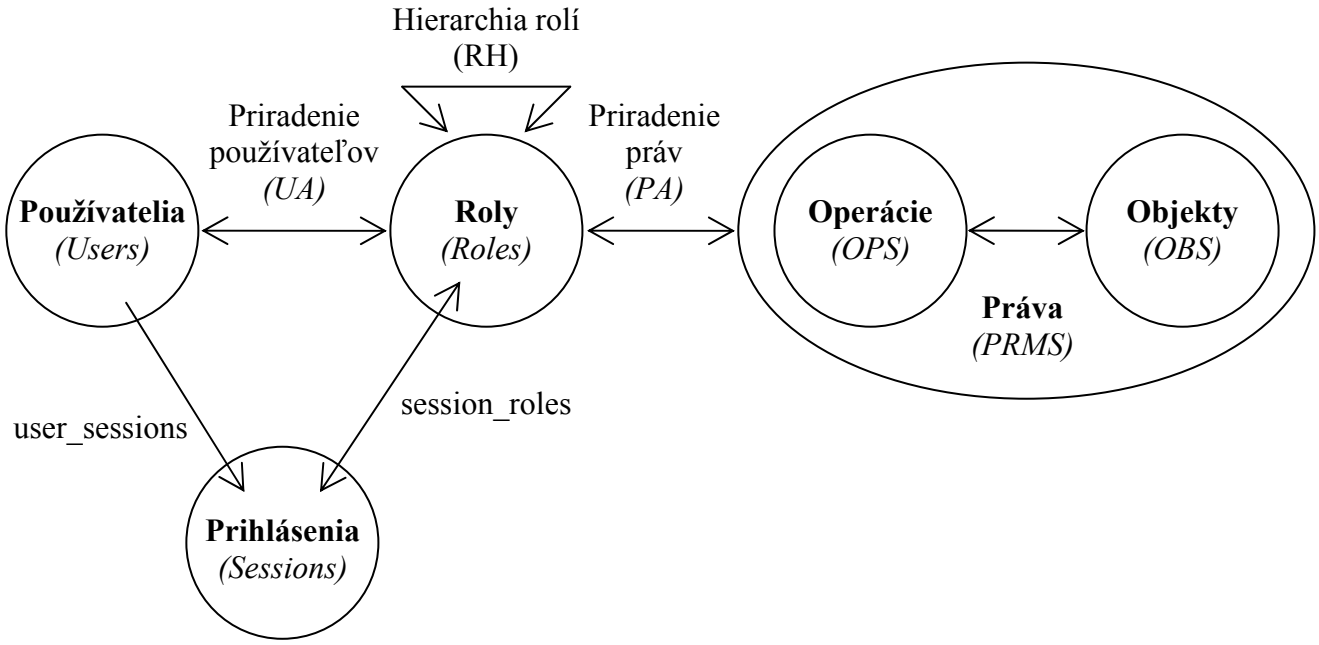
d) Rozšírené prehľadové funkcie

RolePermissions, UserPermissions – vráti množinu práv priradených k danej role alebo používateľovi

SessionRoles, SessionPermissions – vráti množinu aktívnych rolí alebo priradených práv k prihláseniu

RoleOperationsOnObject, UserOperationsOnObject – vráti množinu operácií, ktoré môže daná rola alebo používateľ vykonávať na danom objekte

5.2. Hierarchický model RBAC



Obr. 5: Hierarchický model RBAC (všeobecná hierarchia).

Hierarchický model je pravdepodobne najdôležitejšie a najčastejšie používané rozšírenie základného RBAC modelu. V základnom RBAC modeli môžu mať rôzne roly určitú množinu spoločných a množinu jedinečných práv. Obvyklou požiadavkou je, aby sme každému používateľovi v systéme priradili minimálnu množinu práv, nemôžeme preto tieto roly spojiť do jednej. Prideľovanie práv každej role osobitne je však prácne a môže sneprehľadniť systém. Hierarchický model nám umožňuje spoločné práva priradiť novej role v systéme a ostatné roly nechať tieto práva zdediť.

Hierarchický model pridáva možnosť usporiadať roly do hierarchie, ktorej v matematike hovoríme čiastočné usporiadanie. Pripomenieme si jej definíciu z algebry [5].
Relácia $\circ$ na množine A sa nazýva

a) *reflexívna*, ak pre každé $a \in A$ platí $a \circ a$
b) *antisymetrická*, ak pre každé $a, b \in A$ z $a \circ b \wedge b \circ a \Rightarrow a = b$

c) *tranzitívna*, ak pre všetky *a, b, c* ∈ A z $a \circ b \wedge b \circ c \Rightarrow a \circ c$

Čiastočné usporiadanie ○ množiny A je reflexívna, antisymetrická a tranzitívna relácia (budeme ho označovať $\succeq$).

V relácii A $\succeq$ B nazývame rolu B podriadená a rolu A nadriadená. Nadriadená rola dedí práva podriadenej roly a podriadená rola dedí príslušnosť používateľov nadriadenej roly. Čiastočné usporiadanie hierarchie rolí sa dá zakresliť pomocou orientovaného grafu (hrana grafu smeruje od nadriadenej roli k podriadenej).

Formálne reláciu čiastočného usporiadania definujeme $RH \subseteq ROLES \times ROLES$. Pre roly r_1 a r_2 navyše ďalej platí

$$r_1 \succeq r_2 \Rightarrow authorized\_permissions(r_2) \subseteq authorized\_permissions(r_1) \wedge$$
$$\wedge\ authorized\_users(r_1) \subseteq authorized\_users(r_2),$$

kde $autorized\_permissions(r : ROLES) \rightarrow 2^{PRMS}$ je množina práv priradených danej role a je definovaná nasledovne

$$autorized\_permissions(r) = \{p \in PRMS \mid r' \succeq r, (p, r') \in PA\}.$$

Množina $autorized\_users(r : ROLES) \rightarrow 2^{USERS}$ predstavuje množinu všetkých používateľov priradených k role *r* alebo rolu *r* dediacich. Nahrádza množinu *assigned_users(r)* zo základného modelu. Formálne je definovaná nasledovne

$$authorized\_users(r) = \{u \in USERS \mid r' \succeq r, (u, r') \in UA\}$$

Z definície čiastočného usporiadania vyplýva niekoľko vlastností hierarchie rolí

a) *reflexívna* - zavedením hierarchie žiadna rola nestratí priradené práva (platí len v systéme bez negatívnych práv popísaných v kapitole 7.3)
b) *antisymetrická* - rola A nemôže byť pre rolu B nadriadená aj podriadená
c) *tranzitívna* - rola dedí práva od všetkých rolí, pre ktoré je nadriadená
d) *antisymetrická a tranzitívna*, v grafe hierarchie nesmieme mať cyklus

Podľa štandardu je možné klásť obmedzenia na graf hierarchie. Hierarchie delíme na

a) všeobecné hierarchie – hierarchia je ľubovoľné čiastočné usporiadanie
b) limitované hierarchie – graf hierarchie je najčastejšie strom (koreňom stromu je podriadená rola) alebo otočený strom (koreňom stromu je nadriadená rola), ale môže to byť aj graf s inými obmedzeniami (strom maximálnej hĺbky, ap.). Strom a otočený strom ilustruje príklad na Obr. 6. Napriek tomu, že limitovaná hierarchia nemá všetky možnosti všeobecnej hierarchie, zjednodušuje administráciu systému. Jej použitie má zvyčajne

dôvody nesúvisiace s RBAC systémom (obmedzenia organizácie alebo aplikácie využívajúcej RBAC, atď.).

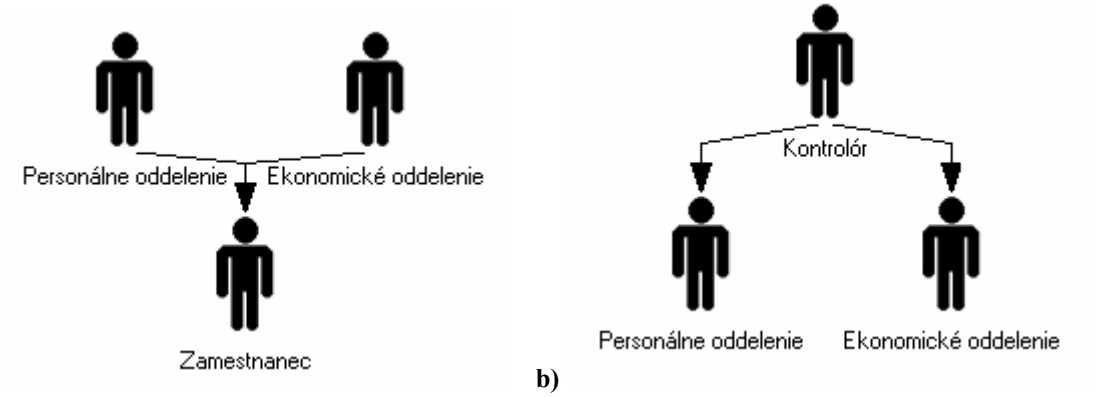
Formálne môžeme limitovanú hierarchiu definovať nasledovne:

a) strom

$\forall r, r_1, r_2 \in ROLES, r \succeq r_1 \wedge r \succeq r_2 \Rightarrow r_1 = r_2$

b) otočený strom

$\forall r, r_1, r_2 \in ROLES, r_1 \succeq r \wedge r_2 \succeq r \Rightarrow r_1 = r_2$



a) **b)**

Obr. 6: Dve z možných usporiadaní rolí v limitovanej hierarchii: a) strom a b) otočený strom.

Vieme teda, že okrem práv, ktoré priamo priradíme danej role môžeme role práva priradiť aj nepriamo tak, že ich zdedí od inej roly. Práva, ktoré sme priradili role priamo môžeme odobrať. O tom, či môžeme role odobrať aj zdedené práva však štandard nehovorí (pozri kapitolu 7.3).

Hierarchický model rozširuje a nahrádza množinu funkcií základného modelu o nasledovné funkcie:

a) Administratívne funkcie

AddInheritance, DeleteInheritane – pridá alebo odstráni zo systému reláciu priamej dedičnosti dvoch rolí

AddAscendant, AddDescendant – vytvorí novú rolu ako priameho predka alebo nasledovníka danej roly

AssignUser, DeassignUser – vytvorí alebo odstráni priradenie roly a používateľa

GrantPermission, RevokePermission – vytvorí alebo odstráni priradenie roly a práva

b) Systémové funkcie

Nepridáva nové funkcie.

c) Prehľadové funkcie

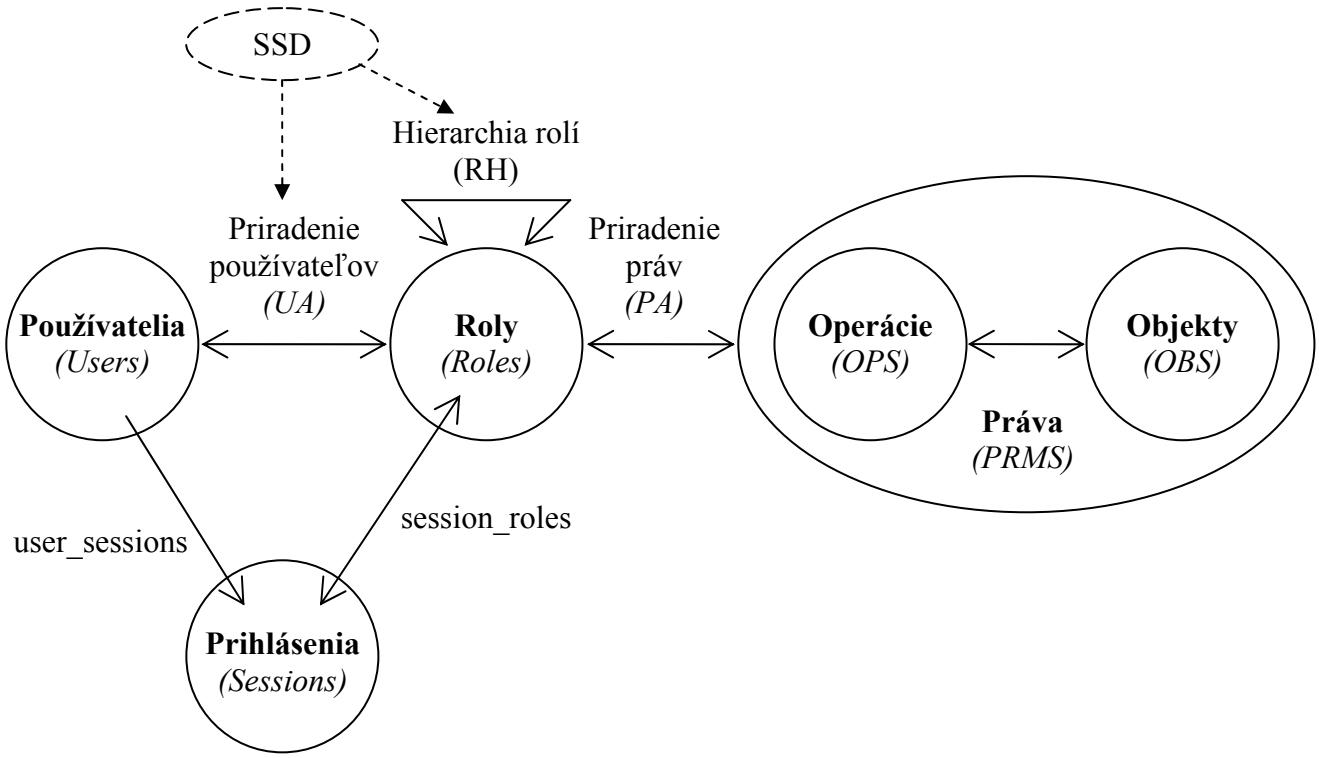
AuthorizedUsers – vráti množinu používateľov priradených a zdedených danou rolou

AuthorizedRoles – vráti množinu rolí priradených a zdedených daným používateľom

d) Rozšírené prehľadové funkcie

Nepridáva nové funkcie.

5.3. Statické vylúčenie práv



Obr. 7: Statické vylúčenie práv nepovolí používateľovi súčasne zastávať viac ako *k* rolí z definovanej množiny.

V niektorých prípadoch môžeme požadovať, aby používatelia nemali právo súčasne pristupovať k určitým objektom (každý pracovník organizácie má právo pristupovať len k časti databázy klientov, čítať časť zmluvy, ap.) alebo vykonávať súčasne niektoré operácie (účtovník vykonávajúci audit vlastnej práce, agent poisťovne vstupujúci súčasne ako klient tej istej poisťovne). Možným riešením takejto situácie je rozdelenie právomocí na viac pracovníkov organizácie. To znamená rozdelenie práv medzi niekoľko rolí v systéme. Ak by však jeden používateľ nejakým spôsobom (napr. zdedením od iných rolí) získal príslušnosť

do všetkých týchto rolí, mohol by opäť vykonávať nežiaduce operácie. NIST štandard rieši túto situáciu definovaním relácie *statického* a *dynamického vylúčenia práv*.

Relácia statického vylúčenia práv (SSD – *Static Separation of Duty*) je definovaná nasledovne

$$SSD \subseteq (2^{ROLES} \times N)$$

Je to teda množina dvojíc *(rs, n)* kde *rs* je množina rolí a *n* je prirodzené číslo a platí $n \geq 2 \wedge |rs| \geq n$. Táto dvojica udáva, že z množiny rolí *rs* nemôže mať používateľ súčasne priradených *n* a viac rolí, formálne:

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} authorized\_users(r) = \phi$$

Statické vylúčenie práv má zmysel používať spolu s hierarchickým modelom, ale aj bez neho.

a) Administratívne funkcie

CreateSSDSet, DeleteSSDSet – pridá alebo odstráni zo systému SSD reláciu rolí

AddSSDRoleMember, DeleteSSDRoleMember – pridá alebo odstráni rolu z množiny vylučujúcich sa rolí

SetSSDCardinality – pre dvojicu *(rs, n)* nastaví *n*, čiže mohutnosť množiny rolí (podmnožiny *rs*), ktorej môže byť naraz priradený jeden používateľ

b) Systémové funkcie

Nepridáva nové funkcie.

c) Prehľadové funkcie

SSDRoleSets – vráti všetky dvojice *(rs, n)* SSD relácií definovaných v systéme

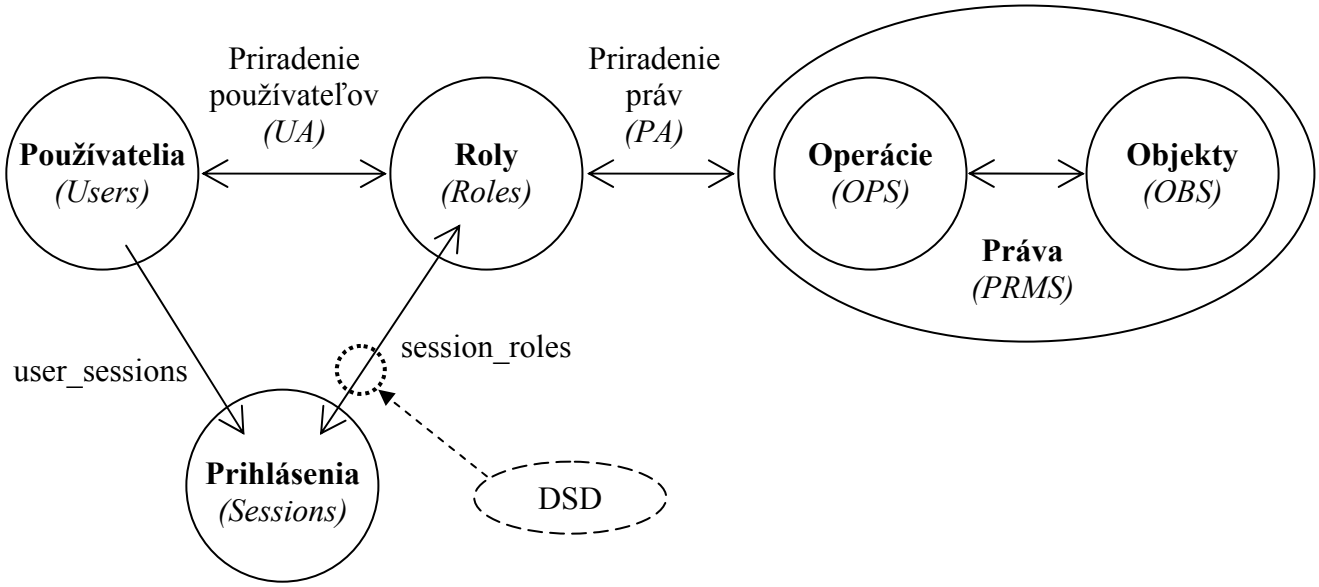
SSDRoleSetRoles – z dvojice *(rs, n)* vráti množinu *rs*

SSDRoleSetCardinality – z dvojice *(rs, n)* vráti *n*

d) Rozšírené prehľadové funkcie

Nepridáva nové funkcie.

5.4. Dynamické vylúčenie práv



Obr. 8: Dynamické vylúčenie práv nepovolí používateľovi počas jedného prihlásenia zastávať *n* a viac rolí z definovanej množiny.

V predchádzajúcej kapitole sme si ukázali možnosti SSD. SSD umožňuje zakázať používateľovi priradiť *n* a viac rolí z určitej množiny. Často je to však príliš silná podmienka, ktorá nadmerne obmedzí práva používateľov. Napríklad ak zakážeme agentovi poisťovne byť súčasne klientom tejto poisťovne, pričom by stačilo zakázať mu podpisovať vlastné zmluvy. NIST umožňuje aj takéto dynamické vylúčenie práv.

Formálne je relácia dynamického vylúčenia práv (DSD – *Dynamic Separation of Duty*) definovaná nasledovne

$\mathrm{DSD} \subseteq (2^{ROLES} \times N)$ je množina dvojíc *(rs, n)*, kde $n \geq 2 \wedge |rs| \geq n$. Táto dvojica udáva, že žiadny používateľ nemôže mať v jednom prihlásení aktívnych *n* a viac rolí z množiny *rs*.

$\forall (rs, n) \in DSD, \forall s \in SESSIONS,\ \forall rs' \in 2^{ROLES}, rs' \subseteq rs, rs' \subseteq session\_roles(s) \Rightarrow |rs'| < n$

Množina *session_roles(s)* predstavuje množinu všetkých rolí, ktoré má používateľ aktívne v prihlásení *s* (používateľ, ktorý vytvoril prihlásenie *s*).

DSD sa teda od SSD líši v kontexte vylúčenia (SSD pracuje v kontexte celého systému, DSD iba v kontexte prihlásenia).

a) Administratívne funkcie

CreateDSDSet, DeleteDSDSet – vytvorí a pridá alebo odstráni dvojicu *(rs, n)* DSD relácie

AddDSDRoleMember, DeleteDSDRoleMember – pridá alebo odstráni rolu z množiny *rs* v dvojici *(rs, n)* DSD relácie

SetDSDCardinality – pre dvojicu *(rs, n)* nastaví *n*, čiže mohutnosť množiny rolí (podmnožiny *rs*), ktorá môže byť naraz aktívna v danom prihlásení

b) Systémové funkcie

Nepridáva nové funkcie.

c) Prehľadové funkcie

DSDRoleSets – vráti všetky dvojice *(rs, n)* DSD relácií definovaných v systéme

DSDRoleSetRoles – z dvojice *(rs, n)* vráti množinu *rs*

DSDRoleSetCardinality – z dvojice *(rs, n)* vráti *n*

d) Rozšírené prehľadové funkcie

Nepridáva nové funkcie.

6. Implementácia RBAC

Hlavným cieľom implementácie RBAC bolo vytvorenie modulu, ktorý by bol použiteľný v ľubovoľnom systéme. Aby však mohli analytici ľahko vidieť vlastnosti tohto modulu a programátori spôsob začlenenia do svojich systémov (taktiež kvôli jednoduchšiemu vývoju) sme modul RBAC implementovali ako modul riadenia práv FTP servera.

Celá aplikačná časť (RBAC model aj používateľské prostredie) bola navrhnutá v prostredí *Borland Delphi 7*. Jazyk Delphi sme si zvolili kvôli jeho jednoduchosti a názornosti (našim cieľom bolo ukázať vlastnosti RBAC). Výhodou Delphi je množstvo existujúcich voľne dostupných komponentov, ktoré sme mohli použiť. Využili sme hlavne dva balíky komponentov

- *Internet Direct (Indy) 10.0.52* – FTP Server, http://www.indyproject.org/
- *JEDI-VCL 3.0* – grafické komponenty, graf hierarchie, atď., http://jvcl.sourceforge.net/

Indy FTP server je vďaka svojej otvorenej architektúre vhodný pre naše využitie. Spĺňa štandard *RFC 959* [12], vďaka čomu je zabezpečená jeho kompatibilita s FTP klientmi. Aj napriek jeho aktívnemu vývoju, ktorý prebieha od roku 2002 sme v ňom našli určité nedostatky a chyby, ktoré nám bránili využívať naplno niektoré funkcie (chyba pri vytváraní logov, problémy so stabilitou, atď.). Samotný FTP server však nie je ťažiskom našej práce, tieto chyby teda neboli pre nás kritické.

Výsledný modul je možné používať buď priamo ako knižnicu (*unit*) v aplikáciách Borland Delphi 7, alebo zabaliť do DLL knižnice. Takto je možné použiť modul vo všetkých moderných programovacích jazykoch. Vďaka dobre definovanému rozhraniu jednotlivých RBAC komponentov je používanie RBAC funkcií bez problémov.

V aplikácii využívame jednoduchú *objektovú databázu* (O-DB), ktorá je vrstvou medzi našou aplikáciou a relačným DBMS (RDBMS) systémom.

6.1. Objektový model

Implementácia RBAC je rozdelená do niekoľkých knižníc:

Pomocné knižnice implementujúce O-DB nad RDBMS:

U_RBAC_sets – knižnica pre prácu s tabuľkami ako množinami

- *TDBSet* – základná trieda umožňuje operácie s prvkami množiny podľa ich identifikátorov.

- *TObjectsSet, TOperationsSet, TRolesSet, TSessionsSet, TUsersSet* – dedia vlastnosti triedy *TDBSet*. Zodpovedajú príslušným tabuľkám v databáze.

U_RBAC_assigments – knižnica implementuje relácie databázy

- *TDBAssignment* – trieda relácie medzi dvoma tabuľkami.
- *TAHierarchy, TAPermissions, TARole_Permission, TASession_Roles, TAUser_Roles, TAUser_Sessions* – dedia vlastnosti triedy *TDBAssignment*. Zodpovedajú príslušným reláciám v databáze.

U_RBAC_database – obsahuje konštanty názvov tabuliek a polí v databáze

Knižnice implementujúce RBAC:

U_RBAC_flat – obsahuje triedu *TRBAC_flat*, ktorej metódy zodpovedajú funkciám NIST RBAC štandardu pre základný model.

U_RBAC_hierarchical – obsahuje triedu *TRBAC_hierarchical*, ktorá je zdedená od triedy *T_RBAC_flat* a pridáva možnosti hierarchie rolí podľa hierarchického modelu NIST RBAC.

U_RBAC_exception – ERBACException je trieda, ktorá spracúva výnimky RBAC systému.

Knižnice podporujúce GUI rozhranie:

U_shapesFactory – obsahuje triedu *TShapesFactory*, ktorá zabezpečuje všetky úlohy spojené s grafickým zobrazením a administráciou hierarchie rolí

U_main – obsahuje hlavné okno aplikácie

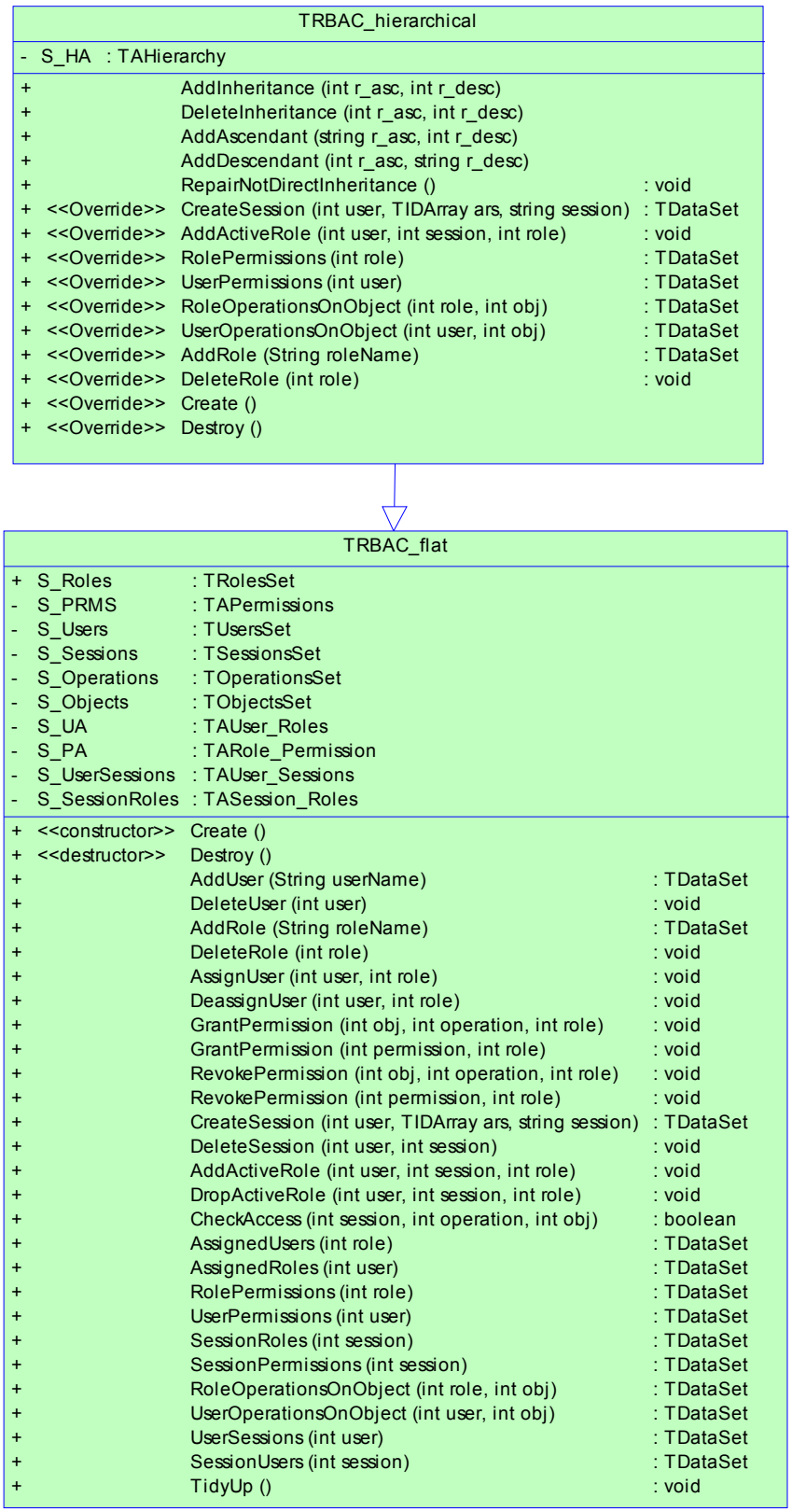
U_frmAddObject, U_frmAddRole, U_frmAddUser – obsahuje okná pridávania objektov, rolí a používateľov

U_frmCustomSelect – pomocné okno so zoznamom, ktorý môže plniť rôzne účely

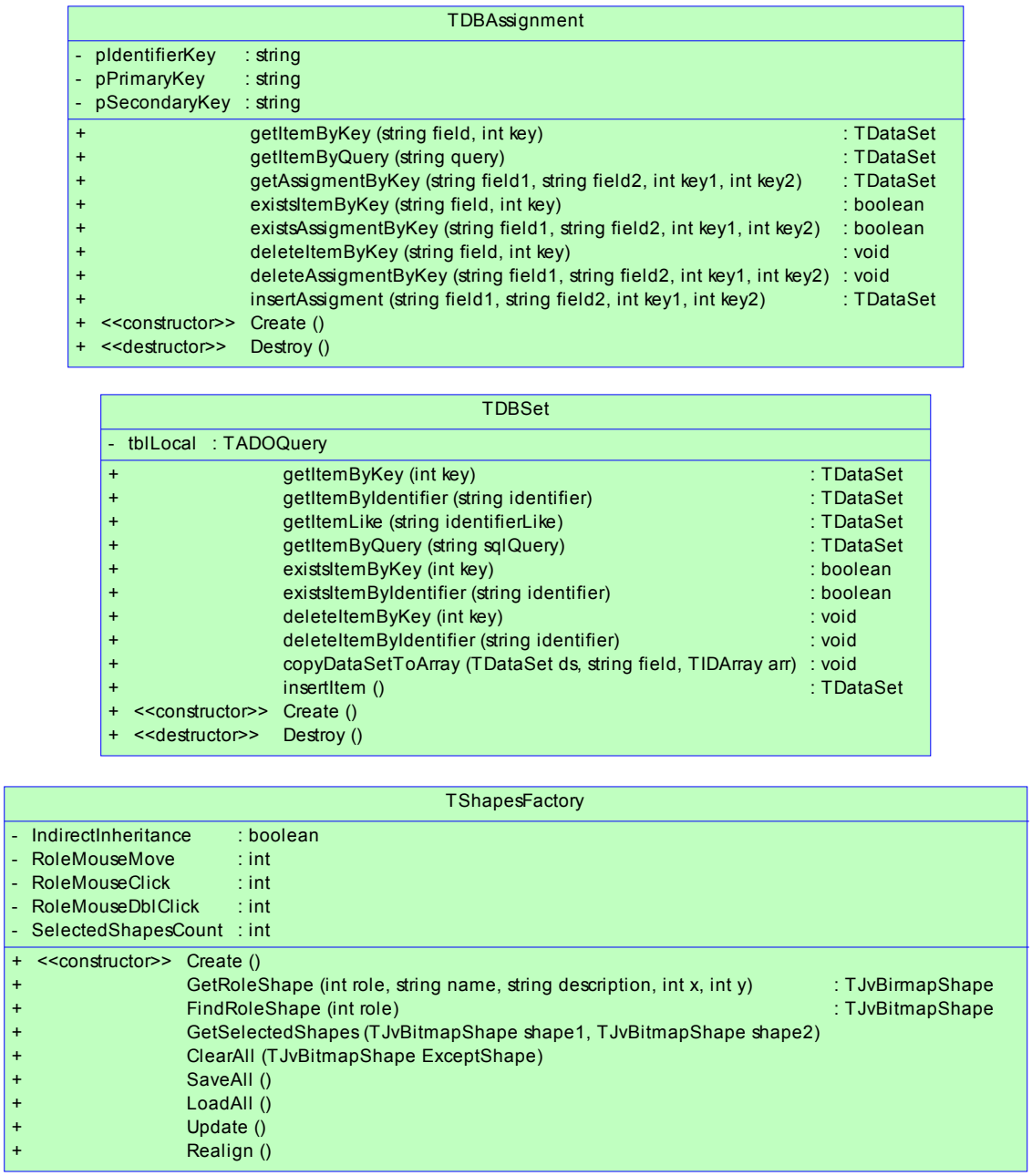
U_DM – DataModule obsahujúci spojenie na databázu

Knižnice podporujúce FTP:

U_session – objekt prihlásenia používateľa na FTP server. Je to jediná knižnica, kde sa spájajú funkcie RBAC a FTP servera v jednej triede.



Obr. 9: Triedový diagram tried RBAC pre základný a hierarchický model.



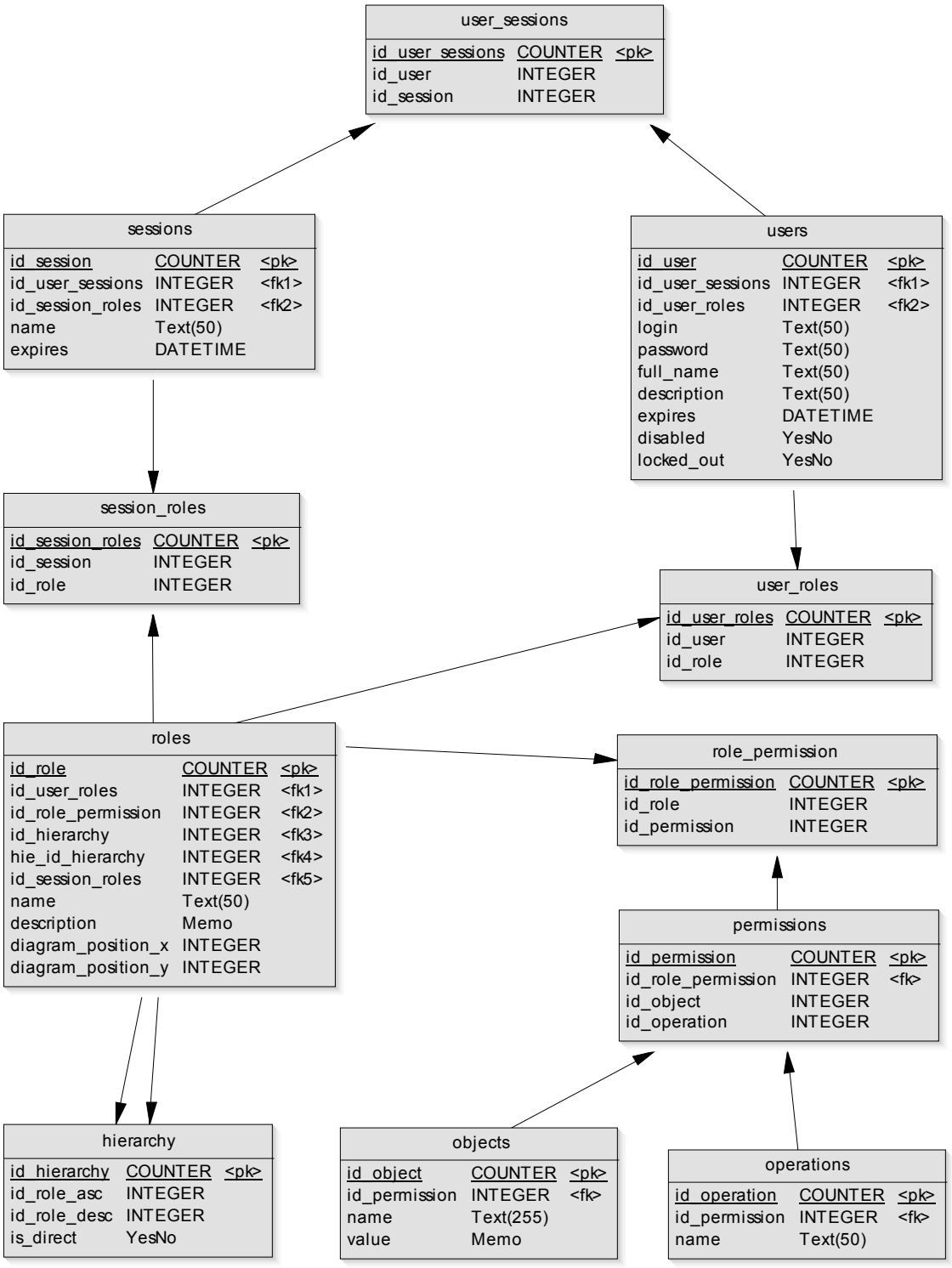
Obr. 10: Triedový diagram pre triedy O-DB a triedy diagramu hierarchie rolí.

6.2. Databázový E-R model

Databázový model priamo odráža štruktúru RBAC systému. Obsahuje tabuľky pre *používateľov, roly, prihlásenia, objekty a operácie*. Do týchto tabuliek je možné pridávať ďalšie polia podľa potreby (pre používateľa čas posledného prístupu, počet neúspešných pokusov o prihlásenie, pre rolu grafické znázornenie v diagrame, atď.). Relácie *n:n* sú riešené pomocou *medzitabuliek*, ktoré neobsahujú iné údaje. Výnimkou je tabuľka *permissions*, ktorá plní súčasne funkciu medzitabuľky aj tabuľky s údajmi.

Tabuľka *hierarchy* kvôli efektívnosti obsahuje informáciu o nadradenosti a podradenosti všetkých rolí v hierarchii (tranzitívny uzáver). Predpokladá sa, že hierarchia rolí bude po vytvorení relatívne nemenná, zatiaľ čo zisťovanie nadradenosti a podradenosti rolí je využívané prakticky v každej funkcii. Vznikne nám určitá redundancia dát v databáze, ale predpokladáme, že graf hierarchie nebude obsahovať príliš dlhé cesty a bude sa skôr vetviť do šírky ako do hĺbky. Vďaka optimalizáciám používaným v DB systémoch je rýchlejšie vyhľadať predpočítané hodnoty ako počítať tranzitívny uzáver znovu. Príznak *is_direct* určuje, či roly v grafe hierarchie priamo susedia alebo ich susednosť vyplýva z tranzitívnosti hierarchie.

Za predpokladu, že umožníme rôznym entitám mať rovnaké mená, relačná schéma je v Boyce-Coddovej normálovej forme (tzn. pre každú netriviálnu závislosť $X \rightarrow A$ platí, že X je *nadkľúč*). V našej implementácii využívame (kvôli jednoznačnej identifikácii entít pre používateľa) rôzne názvy (napr. dve roly nemôžu mať rovnaký názov). Kvôli efektívnosti však použité algoritmy pracujú s číselnými identifikátormi entít. Databáza v našej implementácii je teda iba v 2NF. Tento nedostatok je však skôr teoretického charakteru ako dôsledok zlého návrhu databázy. NIST štandard síce presne neurčuje štruktúru databázy, ale vo funkciách predpokladá využívanie textových názvov entít ako ich identifikátorov. Túto časť štandardu sme nedodržali.



Obr. 11: Diagram fyzického návrhu databázy s vyznačenými tabuľkami, reláciami, primárnymi kľúčmi (*pk*) a cudzími kľúčmi (*fk*).

7. Atribúty RBAC systému

RBAC je rýchlo sa rozvíjajúca a zložitá technológia, preto je ťažké vytvoriť systém vyhovujúci všetkým používateľom. Preto je dôležité definovať určité atribúty RBAC systémov, pomocou ktorých by sa dali popísať požiadavky používateľa, ale aj zhodnotiť výhody a nevýhody použitia konkrétneho RBAC systému pre dané potreby. Sandhu, Ferraiolo a Kuhn [18] spomínajú niekoľko atribútov, ktoré charakterizujú základné vlastnosti RBAC systémov. Ako príklad RBAC systému pre ilustráciu aplikovania atribútov nám poslúži implementácia RBAC FTP servera z kapitoly 6.

7.1. Škálovateľnosť

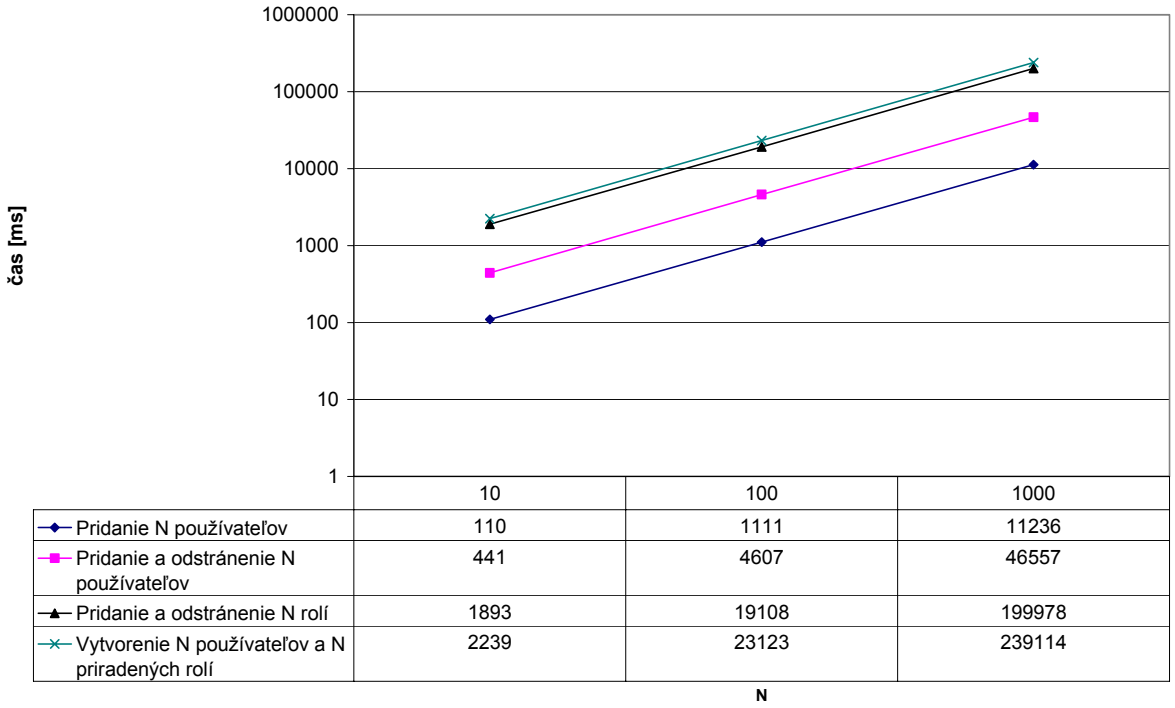
S rastúcou zložitosťou interných i externých počítačových sietí a rastúcim počtom ich používateľov je škálovateľnosť systému (schopnosť efektívne pracovať aj pri zväčšujúcom sa počte entít systému) jednou z najdôležitejších požiadaviek. Dnes existuje mnoho systémov, ktoré dosahujú úroveň základného modelu RBAC (pozri kapitolu 5.1), ktorý je však veľkosťou obmedzený (buď umožňujú využívať iba preddefinované roly bez možnosti pridania ďalších, alebo je počet rolí výrazne obmedzený). Okrem počtu rolí môžeme brať do úvahy počet práv, objektov, možnosť pridávania nových operácií na objekte, atď. Okrem striktného obmedzenia veľkosti systému je dôležité aj zachovanie efektivity vykonávaných operácií pri rastúcej zložitosti systému.

Škálovateľnosť RBAC systému je vždy nutné určovať vzhľadom na prostredie, v ktorom bude RBAC systém prevádzkovaný. Napríklad web server bude zrejme úspešne využívať dáta uložené v relačnej databáze, lebo nebude obsahovať veľké množstvo objektov. Naopak súborový systém obsahuje často desiatky tisícov objektov a preto je dôležité aby množstvo pamäte obsadené systémom pre jeden objekt bolo nízke (napr. Medusa DS9 potrebuje iba 32 bitov na objekt, čo prislúcha možnosti priradiť objektu maximálne 32 virtuálnych svetov, pozri kapitolu 3.2). Škálovateľnosť je často najdôležitejší atribút pri voľbe konkrétnej implementácie.

Naša implementácia RBAC je postavená na relačnom databázovom systéme, množstvo entít systému je teda ohraničené len obmedzeniami použitého databázového servera. Pre prístup k dátam využíva MS ADO objekty, ktoré slúžia ako vrstva abstrahujúca databázový server. Je tak možné využiť prakticky ľubovoľný databázový server, ktorý podporuje ODBC (MySQL, PostgreSQL, Oracle, MS SQL, MS Access, atď.). Štandardne naša implementácia využíva databázový server MS JET a databázu MS Access, ktorý nie je

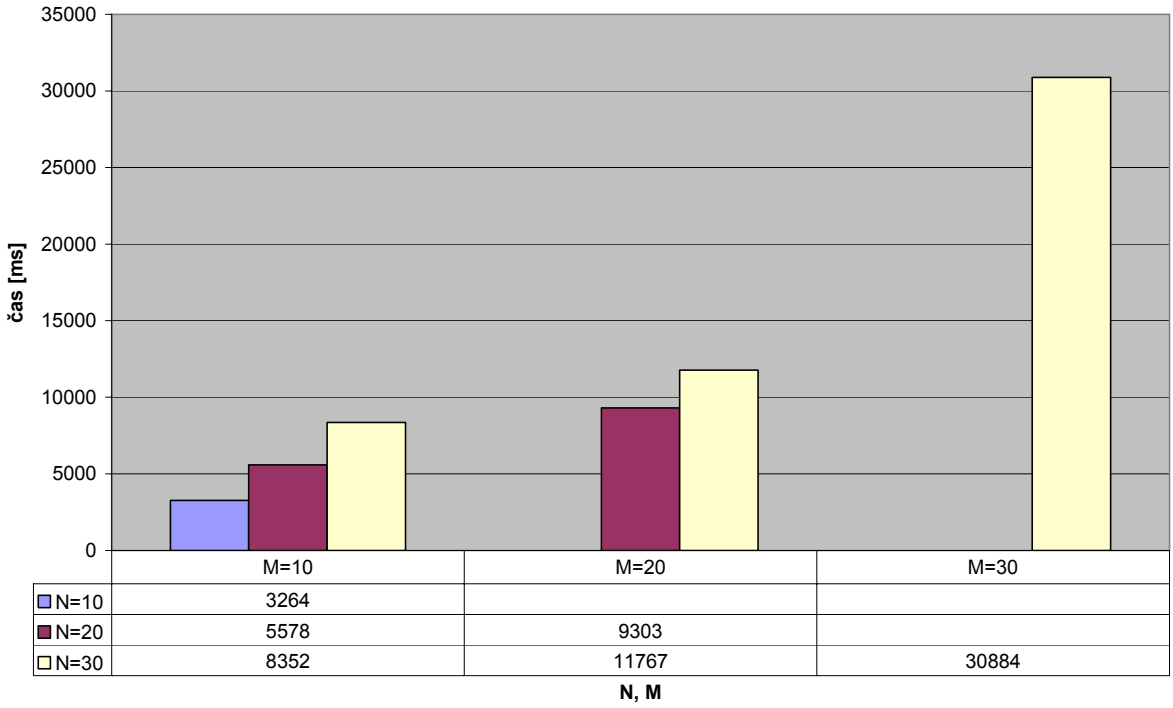
optimalizovaný pre veľké databázy a zložité operácie na nich. Jeho výhoda je však v tom, že vďaka Microsoft Office (ktorého je súčasťou) je nainštalovaný takmer na každom PC. Databáza je uložená v jedinom súbore a netreba ju vytvárať a inicializovať ako v prípade plnohodnotných databázových serverov. Ukazuje sa, že pre jednoduché RBAC systémy postačuje výkon aj takéhoto riešenia.

V grafoch Tab. 4 a Tab. 5 sú uvedené výsledky jednoduchých výkonnostných testov. Boli vykonané na počítači s procesorom Intel Pentium III-M 1000 MHz.



	10	100	1000
Pridanie N používateľov	110	1111	11236
Pridanie a odstránenie N používateľov	441	4607	46557
Pridanie a odstránenie N rolí	1893	19108	199978
Vytvorenie N používateľov a N priradených rolí	2239	23123	239114

Tab. 4: Graf závislosti času (v ms) vykonávania nasledujúcich operácií od *N*: pridanie *N* používateľov, pridanie a odstránenie *N* používateľov, pridanie a odstránenie *N* rolí, vytvorenie *N* používateľov a *N* priradených rolí.



Tab. 5: Čas (v ms) vytvorenia *N* používateľov a *N* rolí priradených k používateľom. Následne vytvorenie kompletnej hierarchie *M* rolí (pre bunky, kde M > N je hodnota prázdna).

Ako vidno na grafe Tab. 4 (graf má logaritmické súradnicové osy), operácie vytvorenia *N* používateľov, vytvorenia a následne odstránenia *N* používateľov, vytvorenia a následne odstránenia *N* rolí sú vykonávané v lineárnom čase v závislosti od *N* – ich počtu. Je teda možné vykonávať ich efektívne aj pri veľkom počte rolí a používateľov v systéme.

Na grafe Tab. 5 je zobrazená závislosť času od počtu vykonaní *N* operácií vytvorenia používateľa a roly priradenej používateľovi a vytvorenia kompletnej hierarchie *M* z vytvorených rolí. Kompletná hierarchia je taká, ktorá vytvorí kompletný graf čiastočného usporiadania rolí (*n+1*-vá rola je nadradenou *n*-tej role v systéme), súčasne je to najzložitejšie usporiadanie rolí v hierarchii v našej implementácii. Táto závislosť už nie je lineárna, čo vyplýva z kvadratického počtu relácii tranzitívneho uzáveru hierarchie. Je teda efektívnejšie pracovať s hierarchiami s nie príliš dlhými cestami v grafe hierarchie.

7.2. Autentifikácia

NIST RBAC štandard nedefinuje, akým spôsobom má byť overená používateľova totožnosť. Napriek tomu je pri implementácii potrebné rozhodnúť, ktorú metódu autentifikácie použiť. Najčastejšou voľbou býva kontrola používateľským menom a heslom, ktoré môžu byť priamo uložené v databáze RBAC systému. Tento spôsob bol zvolený aj pri našej implementácii RBAC FTP servera. Iný spôsob overovania totožnosti môže využívať certifikáty v infraštruktúre s verejnými kľúčmi a pod.

7.3. Negatívne práva

NIST RBAC štandard je založený na prideľovaní prístupových práv rolám a ich dedením v hierarchii rolí. Tieto prístupové práva sa nazývajú tiež *pozitívne* (ak rola získa nejaké právo, už ho nemôže stratiť pridaním novej roly do hierarchie). Napríklad zamestnanec firmy pridelením novej roly účtovníka nestráca práva, ktoré mal v role zamestnanca. Ak mu ich chceme odobrať, musíme mu odobrať rolu zamestnanca.

Naopak *negatívne práva* odoberajú role prístupové práva, ktoré mohla mať. Napríklad ku interným dokumentom organizácie majú prístup zamestnanci všetkých oddelení okrem tých, ktorí sú v role externých pracovníkov. Rovnaký výsledok môžeme vždy dosiahnuť použitím len pozitívnych práv (v tomto prípade napr. rozdelením role zamestnancov na interných a externých, kde obe tieto role dedia práva z role všetkých zamestnancov). Z príkladov vidieť, že systém s negatívnymi právami má zmysel iba pri hierarchickom modeli RBAC.

Použitie negatívnych práv je ponechané na rozhodnutie pri implementácii. Ich nepremyslené použitie však môže viesť k sneprehľadneniu systému, najmä pri použití všeobecnej hierarchie.

Negatívne práva je možné simulovať pomocou pozitívnych tak, že napríklad k právu „povoliť čítať zo súboru“ pridáme právo „zakázať čítať zo súboru“. Ak má používateľ právo povolené, môže vykonávať danú operáciu (čítať zo súboru). Ak ho má zakázané alebo povolené aj zakázané, nemôže čítať zo súboru (pozri Tab. 6). Pomocou tejto logiky je možné využiť existujúci RBAC systém iba pridaním nových práv a zmenou overovania práv pri vykonávaní operácií.

Náš RBAC FTP server priamo nepodporuje negatívne práva.

Právo čítať zo súboru		Operácia povolená
Povoliť	*Zakázať*	
áno	áno	nie
áno	nie	áno
nie	áno	nie
nie	nie	nie

Tab. 6: Simulácia negatívnych práv pomocou pozitívnych práv.

7.4. Štandardizácia práv

NIST RBAC štandard je navrhnutý tak, aby mohol byť použitý ako samostatný modul informačného systému. Práve od požiadaviek systému bude záležať, aké objekty a práva sa budú v RBAC systéme nachádzať. Je preto dobré, ak má implementácia RBAC systému možnosť pridávať ľubovoľné objekty a ľubovoľné práva. To však môže viesť k nedorozumeniam pri špecifikovaní množín práv a interpretácii ich významu medzi rôznymi výrobcami. Štandardizácia práv už však nie je v rozsahu RBAC štandardu a skôr by sa mala uplatniť v danom obore (napr. pre súborové systémy práva čítaj, modifikuj, zapisuj, vykonaj, vypíš obsah adresára, zmaž). Aj v dnešných systémoch tu vládne nekonzistencia. Napríklad súborový systém NTFS umožňuje povoliť alebo zakázať pripojenie dát na koniec súboru oproti EXT3, ktorý takéto právo nemá.

Náš RBAC FTP server podporuje štandardnú množinu práv pre FTP servery, ktoré je možné aplikovať len na virtuálne adresáre (adresáre odkazujúce na skutočný adresár v súborovom systéme OS obsahujúce, ktoré vytvárajú na FTP serveri virtuálny súborový systém). Sú to práva na čítanie, vytváranie, modifikovanie a zmazanie súboru/adresára a výpis súborov a adresárov v adresárovom podstrome daného virtuálneho adresára. Tieto práva odzrkadľujú možnosti FTP protokolu [12].

7.5. Voliteľná aktivácia rolí

NIST RBAC štandard nešpecifikuje, akým spôsobom sa aktivujú roly používateľov. Jediné, čo požaduje je, aby mohol mať používateľ viac aktívnych rolí súčasne. RBAC systémy, ktoré povoľujú iba jednu aktívnu rolu (ktorá sa zvolí pri prihlásení používateľa) teda nespĺňajú tento štandard. Štandard umožňuje aktiváciu všetkých rolí používateľa alebo predvolenú množinu rolí a aktiváciu ďalších prenechať na používateľa.

Pri voľbe metódy aktivácie rolí by sa mala brať do úvahy aj prítomnosť možnosti vylúčenia práv (pozri kapitoly 5.3 a 5.4). Nie v každom prípade je možné aktivovať súčasne všetky roly a naopak nie vždy stačí jedna rola pre vykonanie danej zloženej úlohy.

Naša implementácia RBAC umožňuje zhodne so štandardom postupné aktivovanie rolí. To však v prostredí FTP servera (kvôli obmedzeniam FTP protokolu) nie je možné, lebo mnoho FTP klientov neumožňuje zadávať priamo príkazy FTP serveru (napr. klienti webových prehliadačov) a štandardné príkazy neumožňujú prácu s rolami. Pre zachovanie kompatibility sme teda túto možnosť nechali nevyužitú a aktivujeme všetky dostupné roly používateľa.

7.6. Obmedzenia prístupu

NIST RBAC štandard definuje dve možné obmedzenia prístupu a to statickým a dynamickým vylúčením práv (pozri kapitoly 5.3 a 5.4). Obe tieto obmedzenia predchádzajú vyskytnutiu sa nejakej konkrétnej situácie (napr. používateľ súčasne zastávajúci roly kontrolór a administrátor). Je možné definovať aj iné obmedzenia. Často používanými sú napríklad časové obmedzenie platnosti roly (pracovník smie pristupovať k systému len v pracovných hodinách). Výber týchto obmedzení radikálne ovplyvní možnosti výsledného RBAC systému.

Keďže FTP server neumožňuje postupnú aktiváciu rolí, naša implementácia RBAC neumožňuje definovať dynamické ani statické vylúčenie práv.

7.7. Administrácia RBAC

NIST RBAC štandard neurčuje, akým spôsobom sa riadi prístup k administrácii samotného RBAC systému. V existujúcich systémoch na riadenie prístupu bývajú zvolené rôzne spôsoby určenia práva administrovať systém. Niektoré sa spoliehajú na špeciálnu rolu v systéme, napr. špeciálny tzv. *root* používateľ, ktorý má súčasne právo administrovania systému. Iné obsahujú množinu systémových rolí, ktoré rozdeľujú práva administrovať systém medzi viacerých používateľov. Sú aj také, ktoré dokážu využívajú autentifikáciu používateľov v OS (Microsoft SQL Server).

Ak sa spoliehame iba na používateľské kontá vo vnútri databázy, je nutné zabezpečiť jej bezpečnosť. Je zbytočné chrániť systém uložením hesla administrátora do nechránenej databázy, keď je možné heslo nájsť alebo zmeniť priamo údaje RBAC systému.

V našej implementácii je prístup k administrácii systému obmedzený iba možnosťou zápisu do databázy (zakázaním zápisu využitím práv NTFS súborového systému alebo

nastavením hesla databázy). Pri použití sofistikovanejších databázových systémov ako použitý Microsoft Access je niekoľko možností definovania prístupových práv:

- štandardné prihlásenie používateľa heslom – napr. MySQL, PostgreSQL
- autentifikácia pomocou NTLM – napr. Microsoft SQL Server
- dodatočné formy zabezpečenia (klient/server architektúra a následne ochrana prístupu k databáze pomocou pravidiel firewall-u. Tiež VPN alebo SSH tunel.)

7.8. Odobratie práv role

Spôsob vykonania operácie odobratia práv danej role, môže byť rôzny. Hlavnou otázkou je, či práva odobrať okamžite alebo nie. Najmä pri distribuovaných systémoch nemusí byť jednoduché okamžité vykonanie operácie a mohlo by viesť ku nekonzistencii systému. Kvôli rôznorodosti systémov, ktoré využívajú RBAC systémy, NIST RBAC štandard túto otázku necháva otvorenú. Druhou otázkou, ktorú musí riešiť implementácia je prerušenie prihlásenie používateľa. Operácie v systéme sa často skladajú z transakcií, ktorých prerušenie pred ukončením môže opäť viesť k nekonzistencii systému.

Naša implementácia RBAC systému využíva centrálnu databázu, preto nie je problém ihneď používateľovi zakázať prístup. Povolenie prístupu je kontrolované pred začiatkom vykonávania operácie, preto ak napríklad používateľovi zakážeme čítať zo súboru a používateľ už začal sťahovať súbor, táto operácia nebude prerušená. Ďalšie čítanie zo súboru (aj v tom istom prihlásení) už používateľovi nebude povolené.

7.9. Používateľská prívetivosť

Tento atribút RBAC systémov sa môže zdať nepodstatný, ani NIST RBAC štandard nehovorí nič o používateľskom prostredí. V našej implementácii RBAC FTP servera je zvýšený dôraz kladený najmä na administráciu hierarchie rolí, ktorá je graficky spracovaná a prehľadnejšia ako administrácia cez textové tabuľky dát (pripomína kreslenie UML diagramov). Naša implementácia podporuje zobrazenie priamej i nepriamej dedičnosti v hierarchii rolí. Vďaka tomu, že v tabuľke hierarchie je uložený celý tranzitívny uzáver čiastočného usporiadania rolí je zobrazenie hierarchie rýchle.

7.10. Robustnosť

Počas používania RBAC systému sa používateľom v databáze vytvárajú prihlásenia. NIST RBAC štandard zarážajúco nič nehovorí o tom, čo má systém vykonať po zrútení

systému. Nehovorí ani ktoré operácie sú atomické – nedeliteľné (tzn. mali by byť zahrnuté do transakčného spracovania DBMS).

Naša implementácia zabezpečuje atomickosť funkcií, ktoré definuje štandard (napr. funkcie *AddUser, AddRole,* atď.). Pri chybe v spracovaní takejto funkcie sa obsah databázy vráti do stavu pred jej zavolaním. Pri štarte systému sa vykoná funkcia, ktorá dodatočne zruší prípadné prihlásenia z behu predošlej inštancie systému.

8. Záver

V úvode sme si za cieľ práce položili vytvoriť prehľad existujúcich RBAC systémov, nájsť spoločné atribúty RBAC systémov a implementovať FTP server ilustrujúci výhody RBAC.

V práci sme predstavili niekoľko RBAC systémov. Porovnali sme tri rôzne systémy riadenia práv pre operačný systém Linux a päť rôznych DBMS systémov. Aj napriek tomu, že všetky systémy boli vybrané z rovnakých skupín programov (riadenie prístupu OS Linux a DBMS), ich návrh a možnosti sa značne odlišovali. Podrobne sme popísali štyri komponenty NIST RBAC štandardu (základný model, hierarchický model, statické a dynamické vylúčenie práv) a niektoré z vybraných implementácií sme porovnali s týmto štandardom.

Prvé dva komponenty štandardu (základný a hierarchický) sme implementovali ako modul v prostredí Borland Delphi 7. Tento modul sme použili v FTP serveri ako modul riadenia práv, čím sme ukázali možnosť použitia NIST RBAC štandardu v praxi. Taktiež sme museli vyriešiť niektoré implementačné problémy, ktoré štandard nespomínal (návrh tabuľky dát hierarchie, zabezpečenie robustnosti systému). Za novátorské riešenie administrácie hierarchie rolí považujeme grafickú administráciu hierarchie rolí (podobnú tvorbe UML diagramov). S podobným riešením sme sa nestretli v žiadnom voľne dostupnom systéme a môže si nájsť uplatnenie nie len v RBAC systémoch ale aj v systémoch podobných s RBAC (využívajúcich hierarchiu entít).

Definovali sme množinu hlavných atribútov RBAC systémov. To nám pomohlo popísať vlastnosti našej implementácie a tiež pomocou nich dokážeme porovnať najdôležitejšie vlastnosti iných RBAC systémov. Môžu pomôcť tvorcom nových systémov alebo pomôcť vybrať najvhodnejšiu z existujúcich implementácií. Dúfame, že aj naša implementácia si nájde uplatnenie v systéme, ktorý bude potrebovať riadiť práva a jeho tvorcovia ocenia flexibilnosť NIST RBAC štandardu ale aj nášho modulu.

9. Zoznam použitej literatúry

[1] Department of Defense, Trusted Computer Security Evaluation Criteria, DoD 5200.28-STD, 1985. http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html

[2] D.F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A Role Based Access Control Model and Reference Implementation within a Corporate Internet. http://hissa.ncsl.nist.gov/rbac/jour-rick.ps

[3] D. F. Ferraiolo, R. Kuhn. NIST CSL Bulletin on RBAC, 1995. http://csrc.nist.gov/rbac/NIST-ITL-RBAC-bulletin.html

[4] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security (TISSEC), 2001. http://citeseer.ist.psu.edu/ferraiolo01proposed.html

[5] T. Katriňák, M. Gavalec, E. Gedeonová, J. Smítal. Algebra a teoretická aritmetika 1, Alfa, 2002.

[6] P. Lupták. Porovnanie bezpečnostných implementácií RSBAC a SELinux. http://trip.sk/~wilder/rsbacselinux.pdf

[7] Medusa DS9, domovská stránka projektu. http://medusa.terminus.sk/

[8] Microsoft SQL Server 2000 Books Online, 2000. http://msdn.microsoft.com/library/en-us/startsql/portal_7ap1.asp

[9] NIST Role Based Access Control, domovská stránka projektu. http://csrc.nist.gov/rbac/

[10] A. Ott. The Role Compatibility Security Model, 2002. http://www.rsbac.de/doc/media/rc-nordsec2002.pdf

[11] M. Pikula. Distribuovaný systém na zvýšenie bezpečnosti heterogénnej počítačovej siete, 2002. http://www.ipsec.info/~www/papers/medusa_diplomka.ps

[12] J. Postel, J. Reynolds. RFC 959 – File Transfer Protocol, 1985. http://www.faqs.org/rfcs/rfc959.html

[13] PostgreSQL, dokumentácia. http://www.postgresql.org/docs/

[14] C. Ramaswamy, R. Sandhu. Role-Based Access Control Features in Commercial Database Management Systems, 1999. http://csrc.nist.gov/rbac/RBAC_DBMS_Comparison.pdf

[15] RBAC Cost Estimator. http://csrc.nist.gov/rbac/costs.html

[16] Rule Set Based Access Control, domovská stránka projektu. http://www.rsbac.org/

[17] R. Sandhu. Roles Versus Groups, 1996. http://www.list.gmu.edu/confrnc/rbac/role-group.pdf

[18] R. Sandhu, D. F. Ferraiolo, R. Kuhn. The nist model for role-based access control: Towards a unified standard, 2000. http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf

[19] SELinux, domovská stránka projektu. http://www.nsa.gov/selinux/

Príloha 1: Obsah priloženého CD

bin – spustiteľné súbory aplikácie

∟ *testovacia databaza* – ukážka vybudovanej databázy (je nutné nakopírovať ju do adresára ku exe súboru pred jeho spustením)

prereq – inštalačné súbory knižníc potrebných ku spusteniu aplikácie

praca – prezentácia a PDF dokument tejto práce

source – zdrojové kódy aplikácie

∟ *RBAC* – zdrojové kódy RBAC modulu

UML – obrázky diagramov použité v tejto práci

Príloha 2: Inštalácia RBAC FTP servera

1. RBAC FTP server neumožňuje v konfigurácii zmeniť používaný port, preto sa uistite, že na portoch 20 a 21 nemáte spustený iný FTP server.

2a. Z priloženého CD skopírujte do adresára na pevnom disku súbory *rbaccon.exe* a *data.mdb*. Ak majú atribút *len na čítanie*, tak ho odstráňte (je automaticky pridávaný systémom pre každý súbor na CD).

alebo

2b. Z komprimovaného súboru *rbac.zip* na priloženom CD extrahujte na pevný disk súbory *rbaccon.exe* a *data.mdb*.

3. Spustite aplikáciu *rbaccon.exe*, ak máte s jej spustením problémy, skúste nainštalovať *MDAC* a *JET* z adresára *\prereq*. Ak problémy pretrvávajú, kontaktujte ma na adrese *lubo@tetak.sk*.