Slovenská technická univerzita, Katedra informatiky a výpočtovej techniky
Fakulta elektrotechniky a informatiky, Ilkovičova č. 3, 812 19 Bratislava

Integrácia rôznych bezpečnostných politík do OS Linux

Bc. Marek Zelem

Študijný odbor: Informatika
Ročník: 1.Inž.
Predmet: Diplomový projekt
Vedenie diplomového projektu: Ing. Branislav Steinmüller
Školský rok: 1999/2000

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Študijný odbor: INFORMATIKA
Autor: Bc. Marek Zelem
Diplomový projekt: Intergrácia rôznych bezpečnostných politík do OS Linux
Vedenie diplomového projektu: Ing. Branislav Steinmüller
máj 2000

Cieľom tohto projektu je vytvorenie prostriedkov na začlenenie rôznych bezpečnostných politík do operačného systému Linux. Projekt sa zaoberá analýzou riadenia prístupu na základe rolí (RBAC) ako aj analýzou existujúcich prostriedkov, ktoré môžu umožniť jeho integráciu do operačného systému Linux.

Projekt sa zaoberá aj návrhom implementácie RBAC do operačného systému Linux použitím bezpečnostného systému Medusa DS9. Navrhuje vytvorenie jednotného priestoru mien a definuje 6 všeobecných typov práv. Rieši aj správu bezpečnostného systému RBAC použitím samotného RBAC.

Annotation

Slovak University of Technology, Bratislava
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY
Degree Course: INFORMATICS
Author: Bc. Marek Zelem
Thesis: Integration of various security policies in OS Linux
Supervisor: Ing. Branislav Steinmüller
may 2000

The goal of this project is to create resources for integration of various security policies in operating system Linux. Project analyses the role based access control (RBAC) and existing instruments which can enable integrating RBAC into operating system Linux.

Project also designs the implementation of RBAC into operating system Linux using security system Medusa DS9. It designs unified namespace and defines 6 general permission types. It also solves management of the RBAC system using the RBAC system itself.

Obsah

1 Analýza problému

1.1 Riadenie prístupu na základe rolí - RBAC

Koncept riadenia prístupu založeného na rolách (RBAC) začal vznikať spolu s multi-používateľskými systémami v 70-tých rokoch. Hlavnou myšlienkou RBAC je to, že práva sú priradené rolám a nie používateľom. Používatelia sú priradení jednotlivým rolám. To výrazne zjednodušuje správu. Role sú vytvárané tak aby zodpovedali existujúcim funkciám v organizácii. Ľudia vykonávajúci tieto funkcie sú jednoducho priradení zodpovedajúcim rolám. Rolám sú priradené iba práva potrebné na výkon tej ktorej funkcie. Keď je nejaký človek v organizácii preradený na inú funkciu, tak mu jednoducho stačí zmeniť rolu. Na druhej strane, ak je do systému pridaná nová aplikácia stačí príslušne modifikovať práva rolí. Vo všeobecnosti je potrebné práva rolí meniť menej často ako priradenie používateľov rolám [NIST93]. Tiež je vhodné aby administrátor mohol priraďovať používateľom existujúce role a pritom nemal právo modifikovať práva rolí alebo vytvárať nové role.

1.1.1 Role a skupiny

Mnoho systémov riadenia prístupu umožňuje vytvárať skupiny používateľov. Hlavným rozdielom medzi častou implementáciou skupín a konceptom rolí je, že skupiny typicky predstavujú zoskupenie používateľov a nie zoskupenie práv. Rola je aj zoskupenie používateľov aj zoskupenie práv.

V systéme UNIX je členstvo v skupinách definované v súboroch /etc/passwd a /etc/group. Preto je jednoduché definovať členstvo používateľov v skupinách modifikovaním týchto dvoch súborov. Práva skupín sú však definované pri každom súbore. Je teda výrazne jednoduchšie definovať členstvo v skupine ako práva skupiny. Navyše je priradenie práv skupine silne decentralizované, nakoľko vlastník súboru môže sám určovať skupinu do ktorej súbor patrí, ak je jej členom. Vlastník môže tiež modifikovať práva skupiny na svoj súbor.

Role zodpovedajú dvom charakteristikám:

1) Je približne rovnako ťažké definovať členstvo ako práva.

2) Správa členstva role ako aj správa práv role je centralizovaná.

1.1.2 Štyri triedy modelov RBAC

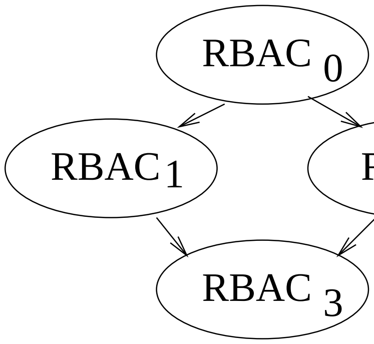
Sú definované štyri triedy modelov RBAC [SAND95].

$RBAC_0$ - Základný model. Predstavuje minimum čo musí obsahovať každý systém riadenia prístupu založený na rolách.

$RBAC_1$ - Rozšírený model. Zahŕňa $RBAC_0$ a pridáva hierarchiu rolí. Teda umožňuje aby jedna rola dedila všetky práva inej role.

$RBAC_2$ - Rozšírený model. Zahŕňa $RBAC_0$ a pridáva množinu obmedzení. Obmedzenia predstavujú logické podmienky, ktoré sú aplikované na všetky časti RBAC.

$RBAC_3$ - Zjednotený model. Zahŕňa $RBAC_1$ a $RBAC_2$.



Vzájomný vzťah medzi triedami modelov RBAC

1.1.2.1 Základný model $RBAC_0$

Obsahuje tri množiny entít:

U - Používatelia (Users)

R - Role (Roles)

P - Oprávnenia (Permissions)

Entita používateľ nemusí predstavovať len človeka, môže to byť aj iná aktívna entita, napríklad bežiaci proces zabezpečujúci nejakú systémovú službu.

Entita oprávnenie predstavuje oprávnenie vykonať nejakú operáciu nad jedným, alebo viacerými objektami v systéme. Oprávnenie je vždy pozitívne. Teda oprávňuje, nezakazuje. Charakter práv silne závisí od systému a od implementačných detailov. Oprávnenie môže byť špecifické, ako napríklad oprávnenie na čítanie časti súboru, alebo všeobecné, ako napríklad oprávnenie na čítanie všetkých súborov patriacich nejakému subsystému.

$RBAC_0$ okrem spomínaných troch množín entít obsahuje aj následovné relácie medzi týmito entitami.

UA - Priradenie používateľov (User Assignment). Je to relácia typu N-N medzi množinou používateľov a množinou rolí, ktorá určuje členstvo používateľov v rolách.

PA - Priradenie oprávnení (Permission Assignment). Relácia typu N-N medzi množinou oprávnení a množinou rolí, ktorá definuje oprávnenia jednotlivých rolí.

S - Prihlásenie (Session). Relácia medzi jedným používateľom a mnohými rolami. Používateľ pri prihlásení aktivuje ľubovoľnú podmnožinu svojich rolí. Táto podmnožina je platná len pre jedno konkrétne prihlásenie. Aktuálne práva predstavujú zjednotenie práv aktivovaných rolí. Používateľ môže byť v jednom čase viackrát prihlásený. $RBAC_0$ tiež umožňuje aby používateľ dynamicky aktivoval a deaktivoval role počas jedného prihlásenia.

1.1.2.2 Hierarchia rolí - $RBAC_1$

Navyše oproti $RBAC_0$ obsahuje reláciu medzi rolami a rolami, ktorá vyjadruje dedenie práv.

Príkladom použitia hierarchie rolí je pracovný vzťah podriadený-nadriadený pracovník. Kde nadriadený je oprávnený robiť všetko to čo jeho podriadený plus má navyše nejaké ďalšie oprávnenia. Niekedy je však potrebné umožniť aby mal podriadený niektoré oprávnenia, ktoré jeho nadriadený nemá automaticky dediť. To sa dá dosiahnuť zriadením takzvanej privátnej role a presunutím práv ktoré sa nemajú dediť do tejto role. Privátna rola bude tiež dediť práva podriadeného. Podriadený bude patriť do privátnej role a teda bude mať navyše niektoré svoje práva, ktoré jeho nadriadený mať nebude.

1.1.2.3 Obmedzenia - RBAC$_2$

Oproti RBAC$_0$ obsahuje RBAC$_2$ navyše množinu obmedzení. Tieto obmedzenia sa vzťahujú na relácie UA, PA a S, a na množiny entít U, R a P. Obmedzenia sú predikáty, ktoré hovoria či sú hodnoty rôznych častí RBAC "akceptovatelné" alebo "neakceptovatelné". Povolené sú len akceptovatelné hodnoty. Z implementačných dôvodov by mali byť výroky obmedzení čo najjednoduchšie, aby sa dali čo najefektívnejšie vyhodnotiť.

Najčastejším obmedzením v RBAC je vzájomné vylučovanie rolí. Teda že jeden používateľ môže byť najviac v jednej role z množiny vzájomne sa vylučujúcich rolí.

Ďalšie časté obmedzenia sú kardinalitné obmedzenia. Rola môže mať stanovený maximálny alebo minimálny počet používateľov ktorý môžu byť jej členmi. Alebo tiež používateľ môže mať stanovený maximálny alebo minimálny počet rolí, v ktorých môže byť.

Ďalším užitočným obmedzením je napríklad, že sa používateľ môže stať členom jednej role A iba ak je už členom role B. Kvôli konzistencii môže byť vyžadované aby role už bolo priradené právo P keď jej je priraďované právo Q.

Podobné obmedzenia môžu byť aplikované nielen na reláciu priradenia používateľov, ale aj na reláciu prihlásenie (session).

Obmedzenia priradenia používateľa vyžadujú aby jeden človek vystupoval iba ako jeden používateľ. Inak by mohol niektoré obmedzenia obísť. Podobne je to s oprávneniami. Povolenie jednej operácie by nemalo byť súčasťou dvoch rôznych oprávnení. Toto musí zabezpečiť správca systému pri jeho konfigurácii

1.1.2.4 Zjednotený model - RBAC$_3$

RBAC$_3$ predstavuje zjednotenie RBAC$_1$ a RBAC$_2$. Teda obsahuje aj hierarchiu rolí aj množinu obmedzení. Pri tejto kombinácii sa vynárajú otázky:

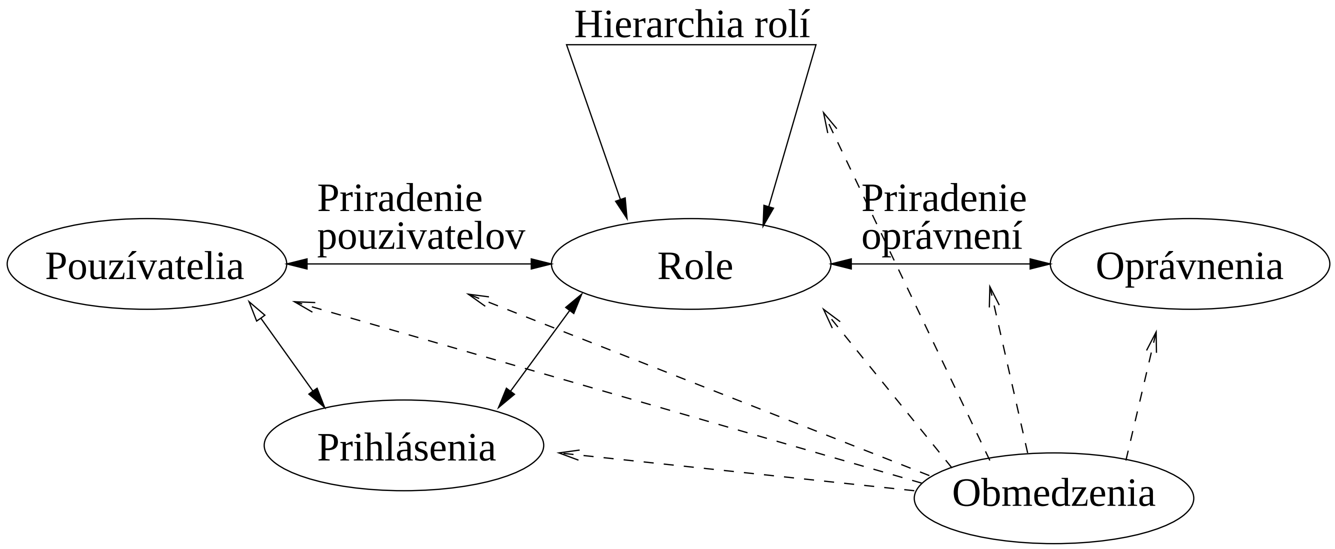
1) Ak rola A a rola B sa navzájom vylučujú a rola C dedí práva role A a rola D

dedí práva role B. Môže byť jeden používateľ v rolách C a D súčasne?

2) Keď má používateľ obmedzený počet rolí, rátajú sa do počtu aj zdedené role?

V prípade že aj zdedené role považujeme za rovnocenné role, tak je potrebné zabezpečiť aby sa zachovalo vzájomné vylučovanie aj pri zdedených rolách. Podobne by mala byť zabezpečená platnosť obmedzení aj pri zdedených rolách.

Kombinácie hierarchie rolí a obmedzení umožňuje decentralizovať správu rolí a hierarchie rolí. Najvyšší bezpečnostný správca môže nastavením vhodných obmedzení limitovať pôsobnosť správcov rolí.



Model RBAC

1.1.3 Správa RBAC

Na správu RBAC sa môžeme pozerať ako na správu relácií. Teda správu priradenia používateľov, správu priradenia práv a správu hierarchie rolí. Okrem správy relácií je potrebné spravovať aj množinu obmedzení.

1.2 Bezpečnostný model OS Linux

Operačný systém Linux je systém typu UNIX, teda jeho bezpečnostný model je veľmi jednoduchý a neposkytuje prostriedky na jeho rozšírenie. Unixový model bezpečnosti je založený na systéme vlastník, skupina, ostatní. Každý súbor v systéme má definovaného vlastníka, skupinu, práva vlastníka, práva skupiny a práva ostatných. Takýto model je značne obmedzený. Navyše vzťahuje sa hlavne na súbory. Aj niektoré ďalšie objekty v systéme majú podobne zabezpečenie. Neexistuje však jednotný model pre všetky objekty. To je spôsobené najmä tým, že systém UNIX vychádzal z abstrakcie, že všetko je súbor. To by znamenalo aj jednotný model bezpečnosti. Žiaľ túto abstrakciu sa tvorcom systému UNIX nepodarilo dôsledne dodržať najmä čo sa týka siete, medziprocesovej komunikácie a procesov, ktoré nie sú zoradené do stromu. Sieť a procesy nemajú definované prístupové práva na základe vlastníka, skupiny a ostatných.

Ďalším faktorom obmedzujúcim bezpečnostný model UNIXu je fakt, že UNIX rozlišuje kvalitatívne iba tri práva. Sú to čítanie, zápis a vykonanie. Na jednej strane je malé množstvo typov práv pozitívne, lebo zjednodušuje a sprehľadňuje administráciu. Na strane druhej však neposkytuje dostatočné možnosti.

Počas behu systému je často potrebné vykonávať spravovanie. UNIX pozná iba jedného správcu s neobmedzenými právami. Tento správca ma tradične identifikačné číslo (uid) 0 a volá sa *root*. UNIX povolí procesu, ktorý beží s rootovskými efektívnymi právami vykonať ľubovoľnú operáciu nad ľubovoľným objektom. Efektívne práva sa používajú na to aby mohol aj bežný používateľ vykonať niektoré privilegované operácie prostredníctvom programu, ktorému root pridelil svoje práva. Takýto program síce beží pod používateľom, ktorý ho spustil, ale má navyše aj rootovské práva. Toto je veľmi nebezpečná vlastnosť, lebo ak chceme niekomu povoliť vykonať nejakú privilegovanú operáciu je potrebné dať programu, ktorý túto operáciu sprostredkuje rootovské práva. To znamená že má právo vykonávať všetky operácie a ochranu pred ich zneužitím musí zabezpečiť sám.

Operačný systém Linux v súčasnosti poskytuje určité vylepšenie modelu systému UNIX o takzvané capability, teda schopnosti, podľa špecifikácie POSIX. Tieto capability predstavujú rozdelenie všemocných práv roota na viacero špecifickejších práv. Capability sa v zásade delia na niekoľko skupín:

1) Capability vzťahujúce sa na súborový systém. Tu patrí napríklad aj právo meniť vlastníka súborov, alebo právo pristupovať k všetkým súborom bez ohľadu na ostatné práva.

2) Capability vzťahujúce sa na procesy. Tu patrí právo posielať signál ľubovoľnému procesu, právo meniť svoj uid a gid, právo meniť svoje capability.

3) Capability vzťahujúce sa na sieť. Tu patrí právo používať TCP a UDP porty menšie ako 1024, právo prijímať a vysielať RAW pakety a rôzne práva administrácie siete.

4) Capability vzťahujúce sa k IPC.

5) Capability na administráciu systému a rôzne špeciálne práva ako napríklad vkladanie modulov do jadra, zmena koreňového adresára pre proces, právo vypínať a reštartovať systém, právo meniť systémový čas...

Tieto capability majú niektoré dobré vlastnosti. Umožňujú, aby aj iný používateľ ako root mohol robiť niektoré, nie veľmi nebezpečné, operácie. Napríklad správca systému, alebo jeden zo správcov, ktorý má samozrejme svoje vlastné používateľské konto a obyčajne nepracuje v systéme s rootovskými právami môže, bez toho aby sa prihlasoval ako root, zrušiť zaťažujúce procesy, ktoré nevlastní, alebo im môže zmeniť prioritu. Tiež je niekedy užitočné, keď takéto právo ma aj nejaký dôveryhodný používateľ, ktorý je často prihlásený a trávi v systéme mnoho času, často viac ako správca.

Niektoré capability dávajú schopnosť ohroziť systém iba v zmysle narušenia jeho normálneho chodu, teda zneprístupnenia služieb. Iné zas poskytujú schopnosť, ktorá sama osebe systém neohrozí, ale umožňuje aby sa dala zneužiť na získanie ďalších práv. To je napríklad právo používať porty menšie ako 1024, alebo právo prijímať a posielať RAW pakety. Medzi capabilitami sú aj také, ktoré predstavujú tak silné schopnosti, že prípadnému narušiteľovi postačí len jedna takáto capabilita. Takou capabilitou je napríklad právo vkladať moduly do jadra alebo právo prístupu a hlavne zápisu do ľubovoľného súboru.

Z uvedeného je zrejmé, že POSIXové capability predstavujú veľmi nerovnomerné rozdelenie práv, čo je veľmi záporná skutočnosť. Ďalšou zápornou vlastnosťou je fakt, že capability predstavujú právo, ktoré sa vzťahuje na všetky objekty. Nie je teda možné mať jednu capabilitu, ktorá platí len na jeden alebo niekoľko objektov. Každá capabilita platí vždy pre všetky objekty.

Praktické použitie capabilít v Linuxe je dosť obmedzené vzhľadom na spomínané záporné vlastnosti. Ďalší problém použitia capabilít je skutočnosť, že ich nepodporujú žiadne systémové programy. Taktiež štandardne neexistuje žiadny používateľsky pohodlný spôsob ich získavania. Existuje niekoľko projektov, ktoré sú zamerané na využitie Linuxovým jadrom podporovaných capabilít.

Jedným z týchto projektov je úprava ELF formátu spustiteľných súborov tak aby umožňovali pri spustení súboru s nastaveným s-bitom pre vlastníka (SUID) určiť ktoré capability má tento program získať. Za normálnych okolností takýto program získal všetky capability. Príkladom použitia je napríklad program ping, ktorý na svoju činnosť potrebuje len jednu capabilitu - právo vysielať RAW pakety. Preto nemusí mať všetky capability. Tým sa znižuje možnosť narušenia systému narušením programu ping.

1.3 Projekty zaoberajúce sa zvýšením bezpečnosti OS Linux

Na umožnenie zmeny bezpečnostného modelu systému Linux je potrebný zásah do jeho jadra (kernelu). Existuje niekoľko projektov zaoberajúcich sa zvýšením bezpečnosti Linuxu. Niektoré z nich však napevno implementujú ich vlastný bezpečnostný model a teda na implementáciu RBAC nie sú vhodné. Na implementáciu RBAC mimo jadra operačného systému je potrebná taká úprava jadra, ktorá umožní nejakému procesu - autorizačnému serveru určovať bezpečnostný model tým, že mu postúpi všetky operácie, ktoré treba autorizovať, na autorizáciu.

1.3.1 Projekty LIDS a VXE

Projekt LIDS (Linux Intrusion Detection System) [LID2000] rieši niektoré konkrétne problémy napevno modifikovaním funkcií jadra operačného systému. Celkovo ale nemá žiadnu jasnú koncepciu. Napriek tomu je tento projekt obľúbený, najmä u laických používateľov operačného systému Linux.

Projekt VXE (Virtual eXecuting Environment) [VXE2000] kontroluje systémové volania. Snaží sa oddeliť procesy, tak aby každý proces mohol robiť iba tie systémové volania ktoré sú potrebné na jeho normálnu činnosť. Na vyhodnocovanie oprávnenia vykonávať volania jadra používa interpreter jazyka lisp priamo v jadre.

1.3.2 Projekt RSBAC

Projekt RSBAC (Rule Set Based Access Control) je zameraný na zvýšenie bezpečnosti operačného systému Linux [RSB2000]. Robí kontrolu bezpečnostne relevantných volaní jadra. Celý bezpečnostný systém sa nachádza v jadre operačného systému. Dobrou vlastnosťou je jeho modulárnosť, lebo umožňuje pridanie rozhodovacích modulov uplatňujúcich rôzne bezpečnostné modely.

Tento bezpečnostný systém postihuje súborový systém, vstupno/výstupné zariadenia, objekty medziprocesovej komunikácie a používateľov. Rozhodnutia o povolení alebo nepovolení prístupu vykonávajú rozhodovacie moduly na základe typu prístupu, objektu na ktorý sa pristupuje a atribútov.

Nevýhodou tohto projektu je, že sa celý bezpečnostný systém nachádza v jadre, takže rozhodovacie moduly nemôžu, ako používateľské procesy, využívať pohodlné služby jadra napríklad na prácu so súbormy.

1.3.3 Projekt Medusa DS9

Projekt Medusa DS9 sa zaoberá podporou autorizačného servera v Linuxe [DS92000]. Vychádza z ročníkového pojektu študentov Mareka Zelema a Milana Pikulu na FEI STU v Bratislave. Projekt mal názov "Zvýšenie bezpečnosti OS Linux" [MEDU99]. Po jeho obhájení ako ročníkového projektu autori pokračovali v jeho dalšom vývoji vo svojom voľnom čase. Neskôr sa k nim pridal aj tretí človek Martin Očkajak, ktorý priniesol nový elán, a tak sa projekt Medusa od jeho obhájenia, ako ročníkového projektu, výrazne vylepšil.

1.3.3.1 Autorizačný server

Od začiatku stál celý projekt Medusa na niekoľkých myšlienkach. Prvá z nich je sledovanie operácii vykonávaných v systéme. Ak v systéme niekto žiada o vykonanie operácie, ktorá je sledovaná, tak túto skutočnosť jadro OS oznámi autorizačnému serveru. Autorizačný server môže zareagovať niekoľkými spôsobmi:

1) Nepovoliť operáciu. To má za následok nevykonanie operácie a následné vrátenie chyby žiadateľovi.

2) Povoliť operáciu. Operácia bude povolená bez ohľadu na štandardné linuxové testovanie práv. Táto možnosť umožňuje aby autorizačný server prevzal plnú kontrolu nad bezpečnosťou a rozhodoval bezprostredne podľa seba.

3) Akceptovať operáciu. V tomto prípade vlastne autorizačný server nezakazuje ani nepovoľuje operáciu. O tom či bude operácia povolená alebo nie rozhodne štandardný linuxový test práv.

4) Prikázať nevykonanie operácie. To znamená, že žiadaná operácia sa nevykoná, ale žiadateľovi sa nevráti chyba, takže sa mu bude zdať, že operácia bola vykonaná.

Pri niektorých operáciách môže autorizačný server uplatniť aj presmerovanie operácie na iný objekt. Zaujímavé je na tom najmä to, že autorizačný server môže o presmerovaní a o objekte na ktorý bude operácia presmerovaná rozhodnúť na základe rôznych informácii o procese, ktorý operáciu požadoval, prípadne aj podľa vlastného vnútorného stavu. Táto schopnosť v praxi našla dobré uplatnenie aj na problémy nesúvisiace s bezpečnosťou, nakoľko sa presmerovanie dá využiť na vytvorenie akéhosi dynamického symbolického odkazu (linku).

Presmerovanie je možné použiť napríklad na vyriešenie problému s tým, že rôzne programy vyžadujú rôzne verzie tej istej knižnice. Prístup na túto knižnicu je možné presmerovať na súbor obsahujúci správnu verziu na základe toho, ktorý program knižnicu požaduje.

Vzhľadom na to, že je potrebné autorizačnému serveru odovzdať čo najužitočnejšie informácie nie je postačujúce zachytávať požiadavky na vykonanie operácií na úrovni systémových volaní. Pri volaní systému sú k dispozícii jedine tie informácie, ktoré poskytol žiadateľ, teda v Linuxe proces. Jadro odovzdá požiadavku autorizačnému serveru až po jej čiastočnom spracovaní, konkrétne po spracovaní argumentov volania systému. Vtedy už má jadro k dispozícii nielen identifikácie objektov zo systémového volania, ale aj interné štruktúry, ktoré ich reprezentujú. V Meduse je ponechaná aj možnosť zachytiť volanie systému. To umožňuje zachytiť každé volanie systému. Takto vlastne existujú dva spôsoby zachytenia požiadavky.

Kvôli dosiahnutiu čo najväčšej efektívnosti môže autorizačný server určiť, ktoré operácie chce sledovať a ktoré nie. Je možné špecifikovať množinu sledovaných systémových volaní pre každý proces zvlášť. Takisto je možné pre každý proces špecifikovať aj sledovanie ostatných operácií. Operáciám, ktoré sú principiálne vykonávané nad nejakým objektom, je možné určiť sledovanie pre každý objekt zvlášť. Takto sa dá jedna operácia postihnúť z oboch strán. Zo strany subjektu, ktorý operáciu vykonáva, a aj zo strany objektu, nad ktorým je operácia vykonávaná. Efektívnosť celého bezpečnostného systému závisí prevažne od schopnosti autorizačného serveru efektívne určiť ktoré operácie, z ktorej strany (objektu či subjektu) a v akej fáze (pri systémovom volaní, alebo pred začatím samotného vykonávania) je potrebné sledovať.

Okrem rozhodovania o tom či bude operácia vykonaná alebo nie môže autorizačný server modifikovať niektoré vlastnosti procesu, ktorý operáciu požaduje. Ide najmä o nastavenia týkajúce sa bezpečnosti ako sledovanie operácií a nastavenia oprávnení procesu.

1.3.3.2 Virtuálne subsystémy

Druhou základnou myšlienkou Medusy sú takzvané virtuálne subsystémy, ktoré slúžia na vzájomné oddelenie tých častí systému, ktoré nemusia byť spojené. To znižuje riziko, že v prípade narušenia jednej časti dôjde aj k narušeniu inej časti. Druhým dôvodom pre vytvorenie mechanizmu virtuálnych subsystémov je skutočnosť, že štandardné linuxové práva túto možnosť neposkytujú v dostatočnom rozsahu. Hlavnou výhodou virtuálnych subsystémov je to, že sa testujú už v jadre. To prináša dve významné vlastnosti:

1) Testujú sa vždy, bez ohľadu na všetko ostatné. Znamená to, že sú neobíditeľné.
2) Sú veľmi efektívne, lebo sa testujú priamo v jadre a nie je potrebné čakať na

autorizačný server.

Príslušnosť objektov k subsystémom určuje autorizačný server, pričom je možné uplatniť mechanizmus dedičnosti. Pri procesoch, ktoré predstavujú subjekty operácií je to zložitejšie. Okrem príslušnosti procesu k subsystémom je možné určiť subsystémy, ktoré proces "vidí", teda vie zistiť existenciu objektov v nich. Ďalej je možné určiť subsystémy, z ktorých môže proces čítať a subsystémy do ktorých môže proces zapisovať. To vytvára širokú škálu možností. Bežným prípadom je umiestnenie všetkých dôležitých systémových programov do jedného subsystému, z ktorého budú môcť procesy len čítať. Získanie práva zápisu do tohto subsystému znamená právo modifikovať systémové programy, prípadne inštalovať nové, čo je dosť zriedkavá činnosť, teda nie je nutné aby toto právo mali bežné systémové procesy.

2 Opis postupu riešenia

2.1 Návrh implementácie RBAC

Navrhnúť systém riadenia prístupu na základe rolí znamená navrhnúť všetky časti RBAC modelu v súlade s už existujúcim systémom. RBAC vyžaduje tri množiny entít: Používatelia, Oprávnenia, Role.

2.1.1 Množina používateľov

V systéme Linux, ako aj vo všetkých systémoch typu UNIX, už množina všetkých používateľov existuje. Nachádza sa v súbore /etc/passwd [HNAT93]. Z pohľadu RBAC používateľmi nie sú iba ľudia ale aj samostatne bežiace programy. V UNIXe je však zvykom aj týmto programom priradiť vlastné UID, teda vytvoriť záznam v súbore /etc/passwd. Napríklad web server obyčajne beží ako používateľ httpd.

2.1.2 Množina rolí

Tak ako množina používateľov musí byť v systéme definovaná aj množina rolí. Kvôli zachovaniu určitých zvyklostí a formátov systémových súborov systému UNIX a teda aj Linux bude najlepšie definovať role v súbore uloženom v adresári /etc. Keďže sa jedná o role, tak ako najvhodnejší názov sa javí /etc/roles. Pre zjednodušenie správy rolí bude najlepšie, keď v tomto súbore nebudú len vymenované všetky role ale budú tam aj bližšie špecifikované. Z hľadiska dodržania známych formátov bude dobré ak bude mať tento súbor podobný formát ako už spomínaný /etc/passwd, alebo /etc/group, či /etc/shadow.

2.1.3 Hierarchia rolí

S množinou rolí silne súvisí relácia hierarchie rolí. Je logické, keď aj táto relácia bude definovaná v súbore /etc/roles spolu so samotnými rolami. Pri každej role bude môcť byť zoznam všetkých podrolí, ktorých práva táto rola dedí. Z hľadiska správy je to korektné, lebo kto má právo spravovať všetky role, vytvárať nové, alebo rušiť existujúce, ten môže mať aj právo určovať ich hierarchiu.

2.1.4 Relácia priradenia používateľov

Ďalšou dôležitou reláciou je relácia priradenia používateľov. Tá by mohla byť takisto súčasťou súboru /etc/roles, podobne ako je to s priradením používateľov skupinám v súbore /etc/group v systémoch typu UNIX. Pre malé systémy s centralizovanou správou je to postačujúce. Pre väčšie systémy, kde je potrebné správu používateľov decentralizovať je vhodnejšie keď je členstvo používateľov v jednotlivých rolách definované v osobitných súboroch. V prípade že je postačujúce spravovať systém RBAC modifikovaním súborov je to užitočné, lebo to zväčšuje možnosti. V takom prípade môžu existovať správcovia rolí, ktorí budú mať právo zápisu do súborov s priradením používateľov rolám, ktoré spravujú. Kompromisným riešením môže byť možnosť v súbore /etc/roles priamo určiť používateľov role, alebo odkázať sa na súbor obsahujúci zoznam používateľov tejto role. Správca jednej role potom bude mať len právo modifikovať tento súbor.

Formát súboru /etc/roles bude následovný:

názov:rid:podrole:popis:používatelia

alebo

názov:rid:podrole:popis:súbor

názov - názov definovanej role.

rid - jedinečné identifikačné číslo tejto role.

podrole - čiarkami oddelený zoznam žiadnej, jednej, alebo viacerých rolí, ktorých práva táto rola dedí.

popis - ľubovoľný reťazec neobsahujúci riadiace znaky ani znak ':'. Vyjadruje slovný popis role. Má len informatívny charakter.

používatelia - čiarkami oddelený zoznam žiadneho, jedného, alebo viacerých používateľov priradených tejto roli.

súbor - absolútna cesta k súboru, ktorý obsahuje zoznam používateľov priradených tejto role. Pole súboru sa od pola používatelia líši práve znakom '/' na začiatku absolútnej cesty k súboru.

2.1.5 Prihlásenie

S používateľmi súvisí ešte jedna relácia - Prihlásenie (Session). Táto relácia je na rozdiel od predchádzajúcich dynamickejšia. Mení sa automaticky počas behu systému. Na jej správu nie je potrebný žiaden človek - správca. Obsah tejto relácie sa modifikuje vždy pri prihlásení nového používateľa, alebo pri odhlásení, alebo pri zmene aktívnych rolí nejakého používateľa. Z toho vyplýva, že o túto reláciu sa musí automaticky starať systém a teda nebude zapísaná v žiadnom súbore, ale bude udržiavaná v pamäti.

V tejto súvislosti je potrebné stanoviť čo je z pohľadu operačného systému prihlásenie používateľa. V Linuxe sa používateľ môže prihlásiť rôznymi spôsobmi. Prihlásenie sa však vždy deje prostredníctvom nejakého externého programu, ktorý síce svojím spôsobom patrí do operačného systému, nie je však súčasťou jadra a preto je problém pre systém detekovať prihlásenie používateľa. Každé prihlásenie používateľa má však jeden charakteristický prvok, a tým je zmena UID. Teda zmenu UID bude systém vždy chápať ako prihlásenie používateľa. Zmena UID sa však niekedy deje aj keď sa žiaden užívateľ neprihlásil, napríklad pri spracovaní prichádzajúcej používateľskej pošty nejakým používateľským programom. Ale aj to sa dá chápať ako prihlásenie používateľa. Pre systém totiž nie je rozdiel v tom, či príkazy vydáva človek sediaci za klávesnicou alebo nejaký samostatný program, ktorý sám aktívne vykonáva rôzne operácie.

2.1.6 Oprávnenia

Každé právo v systéme je principiálne možné rozdeliť na dve časti:

1) typ práva
2) cieľ, na ktorý sa vzťahuje

2.1.6.1 Typy práv

Typ práva vo všeobecnosti predstavuje druh prístupu k objektu, pretože je možné na všetky operácie vykonávané v systéme pozerať ako na operácie vykonávané nad objektami. V oblasti operačných systémov už dlhé roky snahou čo najviac unifikovať všetky objekty v systéme. Vyrazne je to viditeľné na operačnom systéme UNIX, kde bolo snahou takmer všetky objekty unifikovať na súbory. Aj keď sa postupom času nepodarilo UNIXu udržať túto abstrakciu dostatočne dôsledne je možné pozorovať snahu o unifikáciu objektov v systéme Plan 9, ktorý je de facto pokračovateľom myšlienok uplatnených v UNIXe. V nadväznosti na túto snahu o unifikáciu objektov je vhodné snažiť sa aj o unifikáciu operácií nad objektami, ktoré sa v UNIXe žiaľ nepodarilo unifikovať. Výsledkom tohto snaženia je čo najmenšia množina typov operácií nad objektami a teda aj typov práv.

Aby bolo možné nájsť čo najmenšiu použiteľnú množinu práv je potrebné nájsť čo najlepší pohľad na systém. Najdôležitejšou úlohou operačného systému je sprostredkovanie komunikácie medzi objektami. Pričom komunikácia je vlastne tok dát. Objekty v konečnom dôsledku tiež predstavujú dáta. Teda na každý objekt v systéme sa dá pozerať ako na dáta a na každú komunikáciu, nezávisle od toho aký mechanizmus na komunikáciu využíva, ako na tok dát. Z tohto pohľadu vychádzajú štyri typy práv. Je to právo vytvárať objekty, právo rušiť objekty, právo prijímať informácie od objektu a právo posielať informácie objektu.

Okrem týchto štyroch typov práv sú potrebné aj ďalšie dva typy práv, ktoré pokryjú aj všetko to, čo nezapadá do spomínanej abstrakcie. Tými právami sú: Právo nadobúdať nejaký stav. Toto právo určuje cesty v strome jednotného priestoru mien, do ktorých môže proces vstupovať. Pričom právo vstupu do časti reprezentujúcej súborový systém je vlastne pôvodné právo 'x' na adresár zo systému UNIX. Ale právo vstupu do podstromu procesov je v konečnom dôsledku právom spúšťať programy.

Druhým právom je právo riadiť. Pod toto právo spadajú všetky operácie týkajúce sa riadenia iných komunikácii, objektov a systému.

Prehľad všetkých šiestich navrhovaných typov práv je následovný:

1) CREATE - Právo vytvárať objekty a komunikačné kanály. Toto právo zahrňuje napríklad služby UNIXu: create(), connect(), bind(), fork() ...

2) ERASE - Právo rušiť objekty. Toto právo zahrňuje predovšetkým službu UNIXu unlink().

3) RECEIVE - Právo prijímať informácie. Toto právo postihuje predovšetkým čítanie zo súboru, alebo z komunikačného kanála ako je rúra (pipe) alebo socket. Nevzťahuje sa však na službu read(), ale uplatní sa už pri otváraní súboru alebo komunikačného spojenia, tak ako je to v UNIXe zvykom. Práva sa testujú pri otváraní súboru.

4) SEND - Právo posielať informácie. Toto právo tvorí presný protipól k právu RECEIVE. Oproti RECEIVE však postihuje aj zasielanie signálov medzi procesmi, teda službu kill(). V tejto súvislosti by mohlo byť užitočné rozšíriť právo RECEIVE aj o právo prijímať signály, ktoré v klasickom UNIXe nemá obdobu.

5) ENTER - Právo nadobúdať stav. Pod toto právo sa dá zahrnúť napríklad aj zmena UID, spúšťanie programov (exec) ale aj vstupovanie do podadresárov. Toto právo nie je z filozofického hľadiska najvhodnejšie, ale zahrňuje niektoré práva, ktoré v UNIXe existujú.

6) CONTROL - Právo riadiť. Toto zahrňuje všetky špeciálne riadiace práva ako napríklad ptrace(), chmod(), chown() ...

2.1.6.2 Ciele práva

Druhou časťou práva je cieľ na ktorý sa toto právo vzťahuje. Až dvojica typ práva a cieľ predstavuje jedno konkrétne právo. Cieľ pritom môže byť jeden objekt, alebo viac objektov vymenovaných, alebo určených dedičným princípom.

Kvôli prehľadnosti a tým aj ľahšej spravovateľnosti, ako aj kvôli jednoduchému namapovaniu cieľov na virtuálne subsystémy, ktoré budú do značnej miery zabezpečovať niektoré práva, je užitočné definovanie práv rozdeliť na dve časti. Osobitne budú definované všetky ciele, ktoré sa v systéme nachádzajú. To umožní prehľadné rozdelenie objektov v systéme na skupiny - ciele, ktoré sa potom použijú pri definovaní práv. Ciele budú pomenované aby sa s nimi dalo prehľadne a pohodlne pracovať. Určenie cieľov je vlastne moment, ktorý môže spojovať rôzne bezpečnostné politiky a bezpečnostné systémy. Globálne určenie cieľov preto umožní kooperáciu viacerých, aj veľmi odlišných, bezpečnostných politík, ktoré sa takto budú môcť vhodne dopĺňať.

2.1.6.3 Jednotný priestor mien

Aby bolo možné jednotne definovať všetky ciele nezávisle od toho aké objekty systému zasahujú je potrebné zjednotiť identifikáciu objektov v systéme. Inými slovami treba vytvoriť jednotný priestor mien, takzvaný namespace, ktorý zjednotí mená všetkých objektov. Stromová štruktúra súborového systému v UNIXe, ktorý mal pôvodne byť tým zjednocujúcim systémom mien, je na tento účel vhodná. Nezahŕňa v sebe žiaľ iné objekty ako súbory a špeciálne zariadenia. Keďže nie je vhodné zasahovať do už existujúcej konzistentnej štruktúry je potrebné ostatné objekty umiestniť mimo nej. Aby však mali spoločný základ je zavedený nový koreňový uzol. Takto vznikne logická štruktúra, zahrňujúca a hlavne pomenúvajúca všetky objekty v systéme. Táto štruktúra sa v pamäti fyzicky neudržuje, slúži len na jednotnú identifikáciu objektov a tým umožní jednotnú definíciu cieľov práv.

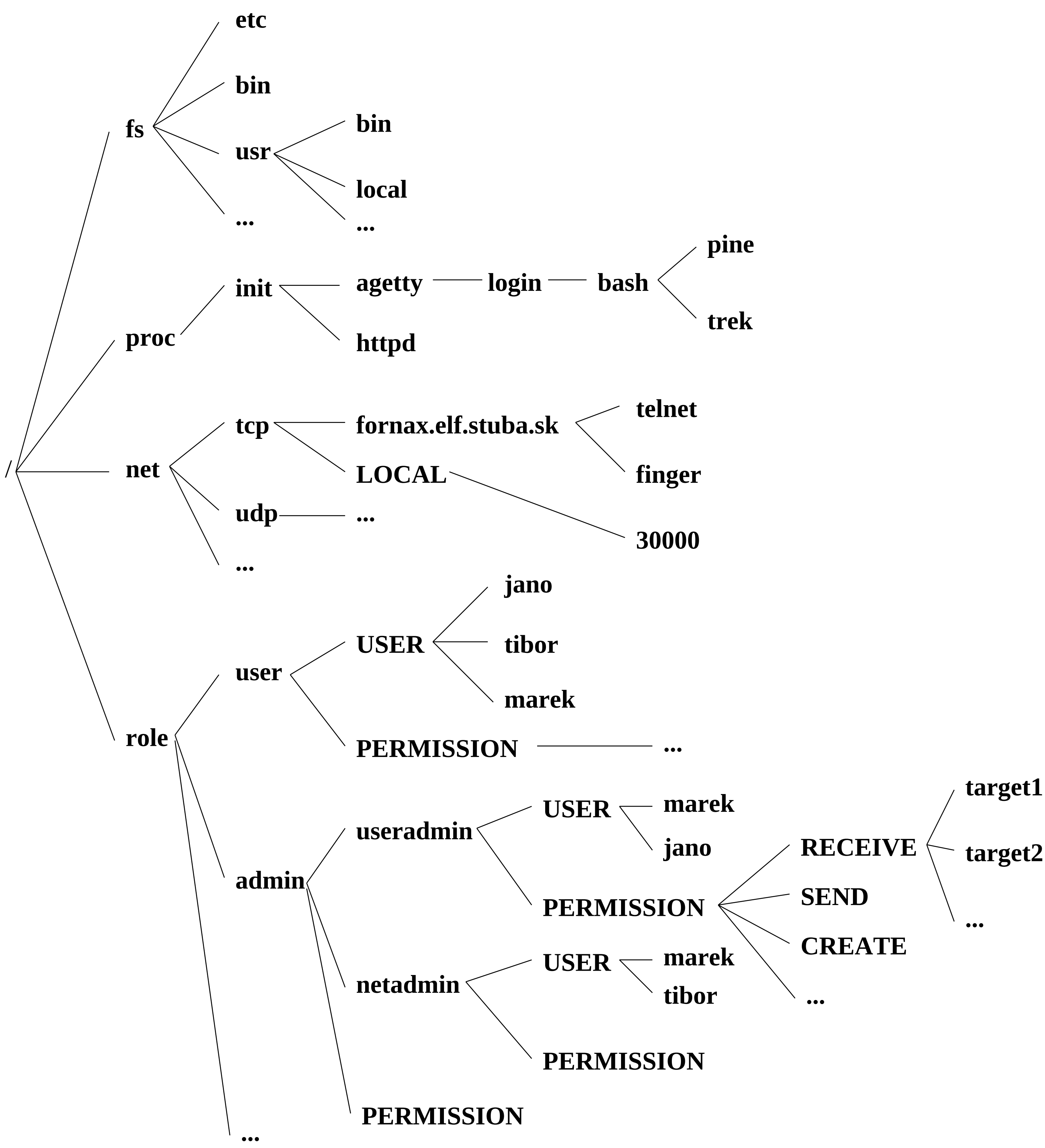
Z nového koreňového uzla vychádza niekoľko vetiev. Jednou z nich je koreňový adresár súborového systému, pod ktorým sa nachádza celý súborový systém. Druhou významnou vetvou vychádzajúcou z globálneho koreňa je podstrom procesov. Na rozdiel od proc filesystému, ktorý sa o zaradenie procesov do súborového stromu pokúšal už skôr, bude podstrom procesov organizovaný hierarchicky. To znamená že všetky procesy nebudú na jednej úrovni, ale budú vytvárať stromovú štruktúru, ktorá bude predstavovať históriu procesov. Cesta od koreňa podstromu procesov až po konkrétny proces bude vyjadrovať cestu procesov ktoré boli od štartu systému spustené, ktoré viedli až k spusteniu procesu na konci cesty.

Procesy nebudú identifikované číslom PID preto lebo PID je priraďovaný dynamicky a pre konfiguráciu bezpečnosti nemá význam. Pri procese je dôležité čo vykonáva, a to závisí od toho aký program je v ňom spustený. Pretože v UNIXe môže byť spustený len súbor existujúci v súborovom systéme, je dobrou identifikáciou procesu meno súboru, ktorý je v ňom spustený.

Týmto pomenovaním by ale vznikol problém pri zápise cesty procesov, keďže jednotlivé uzly podstromu procesov sú od seba oddeľované oddeľovačom a v princípe tým istým oddeľovačom by mali byť oddeľované aj uzly na ceste v súborovom systéme. Tento problém sa dá vyriešiť tak, že namiesto mien súborov sa budú uvádzať mená cieľov smerujúcich do súborového systému. Toto riešenie má oproti iným dve výhody. Prvá je prehľadnosť a druhá je, že proces nemusí byť charakterizovaný len menom jedného súboru, ale niekoľko súborov môže po spustení predstavovať proces s tým istým pomenovaním. Pre špecifikáciu povolených a nepovolených ciest procesov je to užitočná vlastnosť, lebo niektoré aplikácie obsahujú viac spustiteľných programov, pričom je z bezpečnostného hľadiska ich spúštanie rovnocenné. Nevýhodou môže byť zdanlivá anomália, ktorá týmto vzniká, pretože štruktúra globálneho priestoru mien slúži práve na umožnenie definovania cieľov a pritom na definíciu priestoru mien sú potrebné ciele. Toto však nie je problémom pretože strom priestoru mien vlastne fyzicky existovať nebude, ale je len modelom na uľahčenie definícií cieľov. A rekurzívne definovanie cieľov je možné a logické.

Okrem podstromu pôvodného súborového systému a podstromu procesov existuje aj podstrom siete. V tomto podstrome sú uzly predstavujúce rôzne sieťové protokoly, ako napríklad tcp alebo udp, z ktorých vychádzajú uzly predstavujúce adresy iných počítačov, alebo uzol LOCAL, ktorý predstavuje lokálny počítač. Pod týmito uzlami sú uzly predstavujúce protokolové porty. Takto bude možné právami popísať aj právo komunikovať s nejakým vzdialeným počítačom pomocou určitého protokolu s určitým portom, teda službou. Pod uzlom LOCAL sú porty poskytovaných služieb, takže bude možné okrem komunikácie so vzdialenými počítačmi určovať aj právo poskytovať služby, teda čakať na nejakom porte na spojenie.

Ďalším dôležitým podstromom je podstrom *role*, v ktorom sú hierarchicky usporiadané všetky role. Pritom sa jedna rola môže vyskytovať aj na viacerých miestach, lebo jedna rola môže dediť práva viacerých rolí. Pod rolou sa okrem podrolí nachádzajú aj uzly USER a PERMISSION. Pod uzlom USER sa nachádzajú používatelia priradený tejto role. Pod uzlom PERMISSION sa nachádzajú uzly zodpovedajúce typom práv. Pod typmi práv sú uzly predstavujúce ciele práv. Takto sú rolám priradené práva.



Príklad stromu jednotného priestoru mien

2.1.7 Relácia priradenia práv

Poslednou reláciou, ktorú je potrebné do systému riadenia prístupu založeného na rolách zahrnúť, je relácia priradenia práv rolám. Táto relácia je definovaná správcom a preto bude uložená v nejakom súbore. Do súboru je potrebné uložiť aj definície cieľov. Ciele budú uložené v osobitnom súbore /etc/targets, ktorý bude môcť byť prípadne zdieľaný s inými bezpečnostnými politikami. Samotná relácia priradenia práv bude uložená v súbore /etc/permissions, kde budú jednotlivým rolám priradené práva, teda dvojice typ práva a cieľ.

2.1.8 Množina obmedzení

Okrem relácií do systému RBAC patrí aj množina obmedzení. Obmedzenia sú definované správcom a v podstate predstavujú najväčšiu autoritu, lebo môžu obmedziť všetky ostatné časti RBAC. Preto je potrebné, aby boli obmedzenia uložené v osobitnom súbore. Tým vzniká možnosť decentralizovať správu RBAC, ale pritom si právo modifikovať množinu obmedzení môže ponechať jeden hlavný správca. Obmedzenia budú uložené v súbore /etc/constraints.

Obmedzenia sú vlastne predikáty, ktorých platnosť musí bezpečnostný systém zachovávať. V týchto predikátoch je potrebné okrem logických operátorov a kvantifikátorov použiť aj matematické operácie a špeciálne operácie nad objektami vystupujúcimi v systéme, ako napríklad kardinalita množín a relácií.

Vyhodnocovanie obmedzení je potrebné robiť pri každej zmene v systéme, ktorá by mohla zmeniť ich platnosť. Aby bolo možné zabezpečiť optimalizáciu pri ich vyhodnocovaní budú obmedzenia počas inicializácie bezpečnostného systému predspracované a prekonvertované do vnútornej formy.

Obmedzenia budú mať tvar výrazov v ktorých budú vystupovať logické a aritmetické operácie nad entitami, množinami a reláciami RBAC systému. V týchto výrazoch budú môcť byť použité premenné, ktoré budú kvantifikované na začiatku výrazu. Toto uľahčí vyhodnocovanie predikátov.

2.1.9 Správa systému RBAC

Navrhnutý systém, pretože všetky údaju sú uložené v súboroch, umožňuje spravovanie bezprostredne modifikáciou textových konfiguračných súborov. Silnou výhodou toho je možnosť nakonfigurovať systém manuálne. To uprednosťňujú niektorí správcovia. Okrem tejto možnosti bude možné systém RBAC spravovať aj prostredníctvom samotného RBAC systému.

Konfiguračné súbory tak ako aj všetky ostatné súbory sú súčasťou súborového systému, ktorý je pod ochranou RBAC. Takto v konečnom dôsledku systém RBAC chráni sám seba. Pritom niektoré správcovské práva sú vlastne práva zápisu do niektorých konkrétnych systémových súborov.

Súčasťou systému RBAC bude aj mechanizmus jeho správy. Keďže sú role, ich hierarchia, priradenie používateľov a priradenie práv zaradené do podstromu *role* jednotného priestoru mien je možné pomocou neho dobre popísať aj samotný systém RBAC. Ostatné časti RBAC ako množina používateľov a obmedzenia budú spravovaleľné len zmenou súborov, v ktorých sú uložené. Množina používateľov je v systémoch typu UNIX tradične v súbore /etc/passwd. Okrem záznamu v tomto súbore ma používateľ aj svoj domovský adresár a poštovú schránku. Celý proces pridávania a rušenia používateľa je v UNIXe riešený na úrovni systémových programov. S ohľadom na zachovanie kompatibility by nebolo vhodné riešiť správu používateľov iným spôsobom. Množina obmedzení je relatívne stabilná a modifikuje ju väčšinou len jeden správca, takže nevyžaduje zaradenie do stromu.

Právo pridávať alebo rušiť role, priradenie používateľov a priradenie oprávnení sa dá popísať ako právo CREATE a ERASE na nejakú cestu v podstrome *role* jednotného priestoru mien. Tieto práva sa dajú jednoducho a účelne rozdeliť medzi viacerých správcov s obmedzenými právami.

2.2 Sumarizácia

Celkovo je potrebné do systému Linux nainštalovať bezpečnostný systém MEDUSA DS9 a modul do autorizačného servera zabezpečujúci RBAC. Okrem toho je potrebných aj niekoľko systémových programov, ktoré umožnia používateľovi zvoliť si role, v ktorých chce pracovať. Taktiež na správu RBAC budú potrebné ďalšie systémové programy.

Do adresára /etc pribudnú konfiguračné súbory:
roles - množina rolí, hierarchia rolí a priradenie používateľov
permissions - priradenie práv rolám
constraints - množina obmedzení

3 Výsledky riešenia a ich zhodnotenie

Výsledkom riešenia je návrh začlenenia bezpečnostnej politiky RBAC do operačného systému Linux. Návrh počíta aj so začlenením iných bezpečnostných politík. Spojovacím bodom je bezpečnostný systém Medusa DS9, ktorý umožňuje použitie viacerých bezpečnostných politík. Začlenenie iných bezpečnostných politík v najväčšej miere umožňuje návrh zovšeobecnených práv a najmä jednotný priestor mien, ktorý je nezávislý od konkrétnej bezpečnostnej politiky, ale umožňuje jednoduché a hlavne jednotné definovanie práv.

4 Technická dokumentácia

4.1 Požiadavky na systém

Navrhovaný systém vyžaduje na svoju činnosť operačný systém Linux s bezpečnostným systémom Medusa DS9. Do autorizačného servera systému Medusa DS9 je potrebné pridať modul, ktorý bude zabezpečovať bezpečnostnú politiku RBAC.

5 Zoznam použitej literatúry

[SAND95] Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E.: "Role-Based Access Control Models", IEEE Computer 1996.
[NIST93] Ferraiolo, D.F. ; Gilbert, D.M. ; Lynch, N.: "An examination of federal and commercial access control policy needs", NIST-NCSC National Computer Security Conference 1993.
[MEDU99] Zelem, M. ; Pikula, M.: "Zvýšenie bezpečnosti OS Linux", FEI STU Bratislava 1999.
[HNAT93] Hnát, J. : "UNIX V: Příručka správce systému", Grada 1993.
[DS92000] Očkajak, M. ; Pikula, M.; Zelem, M.: "Medusa DS9 security system", 12.5.2000, http://fornax.elf.stuba.sk/medusa/
[RSB2000] "Rule Set Based Access Control (RSBAC) for Linux", 8.5.2000, http://www.rsbac.de/
[LID2000] "Linux Intrusion Detection System (LIDS)", 10.5.2000, http://www.lids.org/
[VXE2000] "VXE - Virtual eXecuting Environment", 1.11.1999, http://www.intes.odesa.ua/vxe/