Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky
Študijný odbor: Informatika

Bc. Róbert Trebula
Vizuálny návrh bezpečnostnej politiky
Diplomová práca

vedúci diplomovej práce: Ing. Branislav Steinmüller
máj 2003

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry.

Bratislava, 12. mája 2003 Róbert Trebula

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky
Študijný odbor: informatika

Autor: Bc. Róbert Trebula
Diplomová práca: **Vizuálny návrh bezpečnostnej politiky**
Vedenie diplomovej práce: Ing. Branislav Steinmüller
Jazyk: slovenský
Kľúčové slová: bezpečnostná politika; riadenie prístupu; vizuálny návrh
máj 2003

Práca aplikuje vizuálne metódy návrhu na oblasť bezpečnostných politík. Na základe analýzy požiadaviek na bezpečnostnú politiku a analýzy spôsobov vizuálneho návrhu prezentujeme jednoduchý diagram ako model bezpečnostnej politiky. Bol navrhnutý a implementovaný vizuálny prostriedok na interaktívnu manipuláciu s takýmito diagramami a aplikovaný na návrh bezpečnostných politík pre niekoľko bezpečnostných modelov. Vytvorený systém je otvorený a umožňuje ďalšie rozširovanie.

Počas návrhu prostriedku – Access Control Designer, bol kladený dôraz na univerzálnosť tohto prostriedku a na možnosť aplikovať vytvorenú politiku na reálne systémy. Jednoduché grafické používateľské rozhranie umožňuje používateľovi abstrahovať od špecifickej syntaxe a sémantiky použitých bezpečnostných mechanizmov a umožní mu zamerať sa na samotnú politiku.

Annotation

Slovak University of Technology, Bratislava, Slovakia
Faculty of Electrical Engeneering and Information Technology
Degree course: Information Technology

Author: Bc. Róbert Trebula
Diploma thesis: **Visual Design of Security Policy**
Supervisor: Ing. Branislav Steinmüller
Language: Slovak
Keywords: security policy; access control; visual design
2003, May

The thesis applies visual methods of design to the domain of access control policies. Requirements on security policies and means of visual design have been analysed. Simple diagram model is proposed for modelling access control policies. An interactive, visually-driven tool to manipulate such diagrams has been designed and implemented. The created system is open and extendable.

While designing the tool – Access Control Designer, accent has been given on versatility of the tool and on the possibility to apply the created policy to real-world systems. Simple graphical user interface allows a user to abstract from the specific syntax and semantics of the security mechanisms involved and allow him to focus on the security policy itself.

Obsah

1 Úvod

Manažment prístupových práv, ako súčasť manažmentu bezpečnosti organizácie je proces komplikovaný, nákladný a náchylný na chyby. Problémom je najmä zložitosť procesu a náročnosť požiadaviek na správne vytvorenie bezpečnostnej politiky pre konkrétnu organizáciu a konkrétny systém.

Bezpečnostné modely, ako napríklad model RBAC vnášajú do problematiky zjednodušenie pri zachovaní požadovaných bezpečnostných vlastností. Najmä však sú tieto modely dostatočne všeobecné, aby mohli byť základom pre univerzálne riešenia vhodné pre aplikáciu v konkrétnych systémoch.

Práca prezentuje takéto univerzálne riešenie – nástroj Access Control Designer, určený na zjednodušenie tvorby bezpečnostných politík. Vizuálnym spôsobom sa pomocou tohto nástroja navrhne politika a následne sa aplikuje na konkrétny systém.

Zjednodušenie návrhu bezpečnostnej politiky je však len jedným z prínosov nástroja. Ako ďalšie môžeme uviesť uľahčenie správy už vytvorených politík, zjednodušenie verifikácie politiky alebo možnosť prezentácie politiky v ľahko pochopiteľnom tvare.

Ako ďalší logický krok v práci na tomto projekte je aplikácia vytvoreného nástroja na reálne systémy. Tým sa vlastne završí smerovanie projektu a splní hlavný cieľ – umožnenie vytvárať bezpečnostné politiky reálnych systémov vizuálnym spôsobom.

Diplomová práca tesne nadväzuje na diplomový projekt, ktorý autor riešil v predchádzajúcich semestroch štúdia (T02). V závere dokumentu (T02) boli uvedené ciele a ďalšie smerovanie projektu. Tieto ciele sa autor snažil splniť počas riešenia zadania diplomovej práce. Ciele sa týkali všetkých etáp vývoja produktu, najmä však išlo o

- cieľ aplikovať vytváraný produkt na reálne systémy,
- získané skúsenosti z používania produktu spätne premietnuť na produkt s cieľom zlepšenia
  - vzhľadom na koncového používateľa produktu,
  - rozhrania pre rozširovanie produktu pre použitie na návrh bezpečnostnej politiky v iných systémoch.

Dokument má nasledujúcu štruktúru: Prvou kapitolou je tento úvod. Druhá kapitola hovorí o bezpečnostnej politike a o jej návrhu. Tretia kapitola hovorí o vybraných bezpečnostných modeloch a mechanizmoch. Štvrtá kapitola

špecifikuje požiadavky na vytváraný softvér. Piata kapitola tieto požiadavky rozanalyzuje. V šiestej kapitole je uvedený návrh vytváraného softvéru, v siedmej je zdokumentovaná jeho implementácia a v kapitole osem je popísané overenie vytvoreného softvérového systému. Kapitola deväť je používateľskou príručkou k vytvorenému systému. Desiata kapitola tvorí príručku pre programátorov, ktorí budú mať záujem použiť vytvorený softvér pre návrh politiky pre nejaké konkrétne prostredie. Jedenásta kapitola tvorí záverečné zhrnutie práce.

Na tomto mieste by som sa chcel poďakovať vedúcemu práce, pánovi Ing. Branislavovi Steinmüllerovi za cenné rady a vedenie projektu. Ďakujem aj svojej rodine a priateľom za ich pomoc a podporu počas celého môjho štúdia.

2 Bezpečnostná politika

2.1 Definície pojmov

Pod pojmom **bezpečnostná politika** (anglicky *Access Control Policy*) budeme rozumieť súbor pravidiel, ktoré určujú, že niektoré subjekty majú niektoré práva k niektorým objektom. Bezpečnostná politika je teda množina trojíc (subjekt, objekt, prístupový mód), pre ktoré platí, že subjekt má právo pristupovať k objektu spôsobom, ktorý opisuje prístupový mód (L71), (BLP76).

Bezpečnostný model tvorí pravidlá pre konštrukciu bezpečnostných politík. Príkladmi bezpečnostných modelov sú prístupová matica, zoznamy prístupových práv, model kapabilít, RBAC model...

Bezpečnostné mechanizmy implementujú bezpečnostnú politiku v konkrétnom prostredí. Sú súčasťou systému, ktorého zdroje bezpečnostná politika chráni.

2.2 Súčasný stav v oblasti návrhu bezpečnostných politík

Súčasný stav v oblasti riadenia prístupu k zdrojom výpočtových systémov je možné charakterizovať v niekoľkých bodoch.

1. Prirodzenou požiadavkou majiteľov týchto systémov je zavedenie a presadzovanie pravidiel riadenia prístupu k zdrojom týchto systémov pri zachovaní a minimálnom obmedzení ich použiteľnosti a efektivity. Vyžaduje sa zavedenie bezpečnostnej politiky na základe všeobecných pravidiel (princíp najmenších privilégií, princíp oddelenia právomocí, logická izolácia používateľov v systéme atď.) a na základe špecifických požiadaviek organizácie a systému samotného.

2. Na formálny opis bezpečnostných politík sú známe bezpečnostné modely, napr. RBAC model[1], prístupová matica, vzory riadenia prístupu a iné teoretické modely, opisujúce riadenie prístupu k zdrojom. Aplikovaním týchto modelov na reálny systém podľa požiadaviek na riadenie prístupu je možné vytvoriť opis bezpečnostnej politiky systému.

3. Na presadenie bezpečnostnej politiky implementujú výpočtové systémy bezpečnostné mechanizmy. Ich konkrétne vlastnosti sú silne závislé od tohoktorého systému. Súbor bezpečnostných mechanizmov na rôznej úrovni (jadro operačného systému, rôzne démony a podporné autentifikačné programy, aplikačné programy, externé bezpečnostné rozšírenia systémov

[1] Bližšie o tomto bezpečnostnom modeli hovorí časť 3.1.

a pod.) umožňujú na väčšine systémov implementovať širokú množinu bezpečnostných politík.

Ak všetky tieto body zhrnieme, vidíme, že vieme presne špecifikovať požiadavky na bezpečnostnú politiku, tieto požiadavky vieme formálne zapísať aplikáciou niektorého bezpečnostného modelu a máme aj prostriedky na to, aby sme túto politiku vedeli implementovať.

2.2.1 Problémy pri tvorbe bezpečnostnej politiky

Problémom pri tvorbe bezpečnostnej politiky je preklad navrhnutej politiky z formálneho jazyka založeného na bezpečnostnom modeli do jazyka bezpečnostných mechanizmov. Inak povedané vytvorenie konfigurácie konkrétnej kombinácie bezpečnostných mechanizmov na základe navrhnutej politiky. Takýto preklad je zväčša netriviálny, pretože:

1. Už aj pri pomerne malých systémoch je rozsah možností a komplexnosť návrhu politiky značná a pre navrhovateľa politiky je pomerne ťažké udržať si prehľad v rozsiahlej konfigurácii bezpečnostných mechanizmov.
2. Spôsob konfigurovania bezpečnostných mechanizmov je špecifický pre každý jeden mechanizmus. Zväčša sa jedná o konfiguračné súbory alebo dialógy v grafickom používateľskom rozhraní. Formát konfiguračných súborov je vzájomne nekompatibilný, v niektorých prípadoch nie je jednoduché ho pochopiť a správne používať.
3. Implementácia bezpečnostnej politiky pomocou kombinácie bezpečnostných mechanizmov je takisto zložitý problém. Ide najmä o zachovanie konzistencie v konfigurácii skupiny bezpečnostných mechanizmov pri nasadení, ale najmä počas údržby bezpečnostnej politiky – pri zmenách týchto konfigurácií.

2.2.2 Existujúce riešenia

Pri analýze existujúcich riešení sme hľadali jednak prostriedky na návrh bezpečnostných politík a okrem toho sme sa zaoberali aj inými nástrojmi, ktoré vizuálnou formou manipulujú s rôznymi objektami.

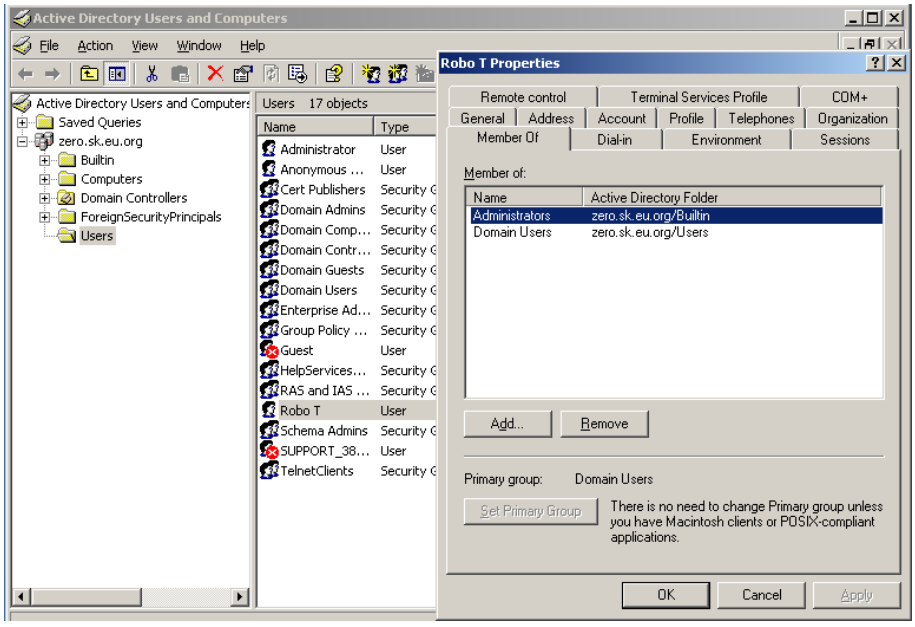
V súčasnosti existuje niekoľko produktov, ktoré sú použiteľné na návrh bezpečnostnej politiky. Niektoré z nich sú:

Rbac/Web (RWEB) je experimentálna implementácia RBAC prístupovej politiky pre použitie vo web serveroch. Obsahuje nástroj na vizualizáciu konkrétnej politiky v trojrozmernom priestore (použitím jazyka VRML). Pomocou tohto nástroja je

možné zobraziť politiky vytvorenú iným komponentom systému za účelom jej kontroly a verifikácie. Nie je však určený na návrh politiky.
Toto riešenie spĺňa požiadavku na vizuálny prístup k navrhovanej politike, nespĺňa však požiadavky na univerzálnosť riešenia a možnosť návrhu politiky.

Role Control Center (FG99) je nástroj na návrh RBAC politiky a jej následnú aplikáciu na cieľový systém. Forma, v akej samotný návrh prebieha sú dialógové okná grafického používateľského rozhrania.
Tento produkt spĺňa požiadavky na univerzálnosť a aplikovateľnosť navrhnutej politiky na rôzne systémy. Jeho používanie však nie je v pravom zmysle slova vizuálne.

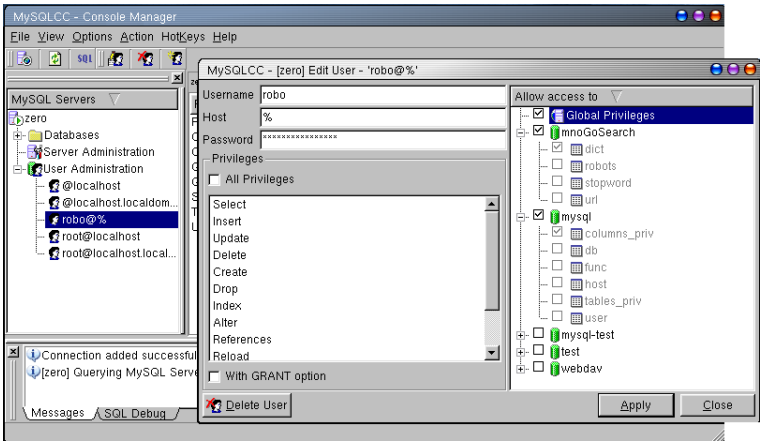
Správa používateľov, skupín a skupinových politík OS Windows je skupina nástrojov na riadenie prístupu k rôznym zdrojom poskytovaným operačnými systémami radu NT, 200X a XP. Týmito nástrojmi sa formou okien v grafickom používateľskom prostredí (príklad na obrázku 1) dá navrhovať, analyzovať a aplikovať politika riadenia prístupu pre tento operačný systém.



Obrázok 1: Správa používateľov v OS Windows

Tieto nástroje sú špecializované na konkrétny operačný systém, teda nie sú univerzálne. Vizuálna stránka nástrojov je obmedzená na objekty typu ikona, dialóg, strom objektov a podobne.

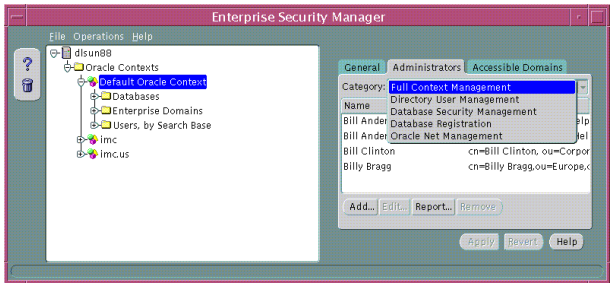
MySQL Control Center (MYSQL) je nástroj na správu databázových serverov MySQL.



Obrázok 2: MySQL Control Center

Obsahuje aj prostriedky na riadenie prístupu používateľov k objektom databázy. Systém pracuje v grafickom používateľskom rozhraní (typická pracovná plocha je na obrázku 2) a je špecializovaný pre konkrétny systém.

Oracle Enterprise manager (ORACLE) je nástroj na správu databázového servera Oracle.

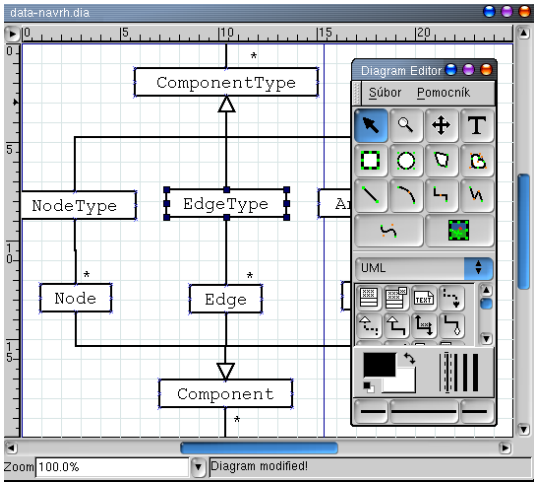


Obrázok 3: Oracle Enterprise Manager

Podobne ako v prípade nástroja MySQL Control Center ide o univerzálny nástroj na manažment databázových serverov, v tomto prípade pre Oracle. Používa grafické používateľské rozhranie (na obrázku 3).

Spomedzi vizuálnych grafických editorov a modelovacích nástrojov bol podrobnejšie preštudovaný jeden nástroj.

Dia je voľne šíriteľný nástroj na tvorbu diagramov. Jedná sa o všeobecný program a význam diagramov neinterpretuje. Existuje však rozhranie, ktoré umožňuje programovať moduly programu dia pre použitie konkrétnych typov objektov a vzťahov (príklad na obrázku 4).



Obrázok 4: Dia

Niektoré schopnosti tohto prostriedku boli zahrnuté do požiadaviek pre program ACDesigner. Jedná sa najmä o možnosti manipulácie so skupinami objektov naraz, kontextové menu pre každý objekt, zobrazovanie vzťahov medzi objektami pomocou spojníc (čiar). Dia však neumožňuje spätnú väzbu od programu k používateľovi – program nekontroluje správnosť používateľových akcií, nevie ho upozorniť na prípadné chyby. Neumožňuje ani vykonávať niektoré funkcie, ktoré by akcelerovali analýzu vytvoreného modelu (napr. vyznačenie všetkých objektov, ktoré sú spojené s nejakým objektom a pod.)

2.3 Životný cyklus bezpečnostnej politiky

Životný cyklus bezpečnostnej politiky, ako je načrtnutý v (CD99) sa skladá z nasledujúcich etáp:

Návrh Identifikácia chránených zdrojov, ich používateľov, identifikácia možných rizík. Vytvorenie politiky ako pravidiel na riadenie prístupu k zdrojom.
Access Control Designer má za úlohu akcelerovať práve túto etapu životného cyklu politiky a preto sa jej venuje aj celá časť 2.4.

Presadzovanie V tejto fáze sa vykonáva prístup subjektov bezpečnostnej politiky k objektom riadený pomocou pravidiel bezpečnostnej politiky.
Počas tejto pracovnej fázy je cieľom nielen dodržiavanie pravidiel politiky a zabránenie neoprávnenému prístupu k zdrojom, ale aj monitorovanie

oprávnených prístupov a pokusov o neoprávnený prístup k zdrojom.
Analýza dát získaná pri monitorovaní je podkladom pre fázu revidovania politiky.

Revidovanie Overenie funkčnosti vytvorenej a používanej politiky riadenia prístupu. V tejto fáze sa identifikujú nutné zmeny v pravidlách politiky.
Pretože systém, o bezpečnosť ktorého sa staráme sa vyvíja, musí tieto zmeny reflektovať aj bezpečnostná politika.
Zmeny v pravidlách politiky majú charakter jednak drobných – operatívnych – zmien (napr. pridanie používateľa, priradenie role používateľovi a pod.) ale niekedy sú potrebné aj zásadné zmeny v celej štruktúre politiky (spôsobené napríklad zmenami organizačnej štruktúry organizácie). Aj v tejto etape je možné použiť prostriedok ACDesigner na podporu týchto činností.

2.4 Návrh bezpečnostnej politiky

Príkladom procesu návrhu bezpečnostnej politiky môže byť proces návrhu politiky založenej na RBAC modeli. Pre návrh bezpečnostnej politiky založenej na RBAC modeli je vhodné postupovať takýmto spôsobom:

1. Identifikovať základné komponenty bezpečnostnej politiky – subjekty, objekty a typy prístupu.
2. Identifikovať roly v systéme na základe analýzy hierarchie a organizačnej štruktúry organizácie.
3. Vytvoriť priradenia používateľov do rolí a priradenie oprávnení rolám na základe analýzy potrieb organizácie.
4. Identifikovať obmedzenia v pravidlách riadenia prístupu (statické a dynamické oddelenie právomocí, obmedzenia kardinality a podobne).
5. Verifikovať vytvorenú politiku. Prihliadať najmä na princíp najmenších privilégií (používateľ by mal mať najmenšiu možnú množinu oprávnení potrebnú na výkon jeho práce).

Tieto fázy sa môžu navzájom prelínať, prípadne sa môže aj zameniť poradie niektorých z nich, vracať sa k činnostiam v predchádzajúcich fázach a podobne. Takéto postupy však zväčša znižujú efektivitu návrhu politiky.

3 Bezpečnostné modely a mechanizmy

Na demonštrovanie možností vytváraného produktu boli vybrané tri rôzne bezpečnostné modely. Táto trojica zahŕňa jednak všeobecný bezpečnostný model (model RBAC), ďalej aplikovaný model (model operačného systému Unix) a aj konkrétnu implementáciu založenú na vlastnom modeli bezpečnosti (riadenie prístupu v MySQL serveri).

3.1 Bezpečnostný model RBAC

Bezpečnostný model RBAC je v súčasnosti najznámejší referenčný model bezpečnostnej politiky. RBAC bezpečnostný model je opísaný v (SCFY95)¸ (FCK95)¸ (SFK00).

3.1.1 Princíp a komponenty RBAC

Riadenie prístupu založené na rolách (Role Based Access Control-RBAC) je koncept modelu riadenia prístupu. V politike založenej na RBAC modeli sú oprávnenia združené s rolami a používatelia sú zadelení ako členovia príslušných rolí. Tento prístup zjednodušuje manažment prístupových práv a vnáša doň cennú úroveň abstrakcie.

Základný koncept rolí je takýto (SFCY95): priradiť oprávnenia na základe funkčných rolí v danej organizácii a potom zaradiť používateľov do príslušných rolí. V RBAC bezpečnostnej politike sa rozhodovanie o prístupe k zdrojom vykonáva na základe rolí, v ktorých vystupujú jednotliví používatelia. Roly sa zakladajú pre rozličné pozície v organizácii a používatelia sa zaraďujú do rolí na základe ich pracovného zadelenia - zodpovednosti a kvalifikácie. Pretože roly v jednej organizácii sú relatívne stabilné a častejšie sa mení príslušnosť používateľov k rolám, RBAC výrazne znižuje komplexnosť, náklady a výskyt omylov administrátora pri správe prístupu k zdrojom. RBAC umožňuje centralizovať správu prístupu, pretože príslušnosť používateľov k rolám aj príslušnosť oprávnení k rolám je možné ovládať centralizovane. Tým je zabezpečená centralizovaná správa celej prístupovej politiky, na rozdiel od koncepcie skupín používateľov (napríklad v operačnom systéme UNIX). Komponenty RBAC sa môžu upravovať a vyvíjať spolu s vývojom samotnej organizácie a to je ďalšou výhodou RBAC.

3.1.2 Vzťahy medzi komponentmi RBAC

Relácie používateľ-rola a rola-oprávnenie sú základom celej koncepcie RBAC. Jeden používateľ môže byť členom viacerých rolí, v každej role môže vystupovať viac používateľov. Jedna rola môže mať priradených viac oprávnení, každé

oprávnenie môže byť priradené k viacerým rolám. *Používateľ* je v tomto modeli chápaný ako osoba. Túto entitu je však možné chápať aj všeobecnejšie a priradiť k nej aj niektoré samostatné procesy (napr. démony), ktoré vystupujú s vlastnou identitou. *Oprávnenie* je prístupové právo k jednému alebo viacerým zdrojom v systéme. Význam *oprávnenia* závisí od konkrétneho systému implementujúceho mechanizmy pre RBAC. Každý systém chráni objekty na úrovni abstrakcie, ktorú sám implementuje.

Hierarchia rolí je prirodzený spôsob štruktúrovania rolí, ktorý rešpektuje štruktúru organizácie - hierarchiu autority a zodpovednosti pozícií v organizácii. Ak je rola nadradená inej role, dedí od nej všetky jej právomoci. Používateľ - člen nadradenej roly môže aktivovať akúkoľvek podmnožinu rolí, od ktorých táto rola dedí.

3.1.3 Obmedzenia

Obmedzenia sú predikáty, ktoré určujú, ktoré kombinácie hodnôt komponentov RBAC modelu sú povolené a ktoré nie. Obmedzenia sú dôležitým prvkom RBAC, pretože umožňujú napríklad zabezpečenie vzájomného vylučovania rolí pre politiky vyžadujúce oddelenie právomocí.

3.1.4 Formálny model RBAC

- U – množina používateľov; $u = |U|$ - počet používateľov
- R – množina rolí; $r = |R|$ - počet rolí
- P – množina oprávnení; $p = |P|$ - počet oprávnení
- UA – relácia príslušnosti používateľov k rolám $UA \subset U \times R = \{(u_i, r_j), ...\}$
- PA – relácia príslušnosti oprávnení k rolám $PA \subset P \times R = \{(p_k, r_l), ...\}$
- RH – relácia hierarchie rolí $UA \subset R \times R$
- *obmedzenia*, ktoré určia, ktoré konfigurácie politiky sú zakázané
- *Hierarchia* je relácia $RH \subset R \times R$. Ak dvojica $(r_1, r_2) \in RH$, platí, že rola r_2 je priamym nasledovníkom role r_1 v tejto hierarchii. Reflexívny a tranzitívny uzáver RH^* relácie RH určuje dedičnosť oprávnení medzi rolami. Ak dvojica $(r_1, r_2) \in H^*$, platí, že rola r_2 dedí všetky oprávnenia role r_1.

3.2 Operačný systém Unix

Operačné systémy typu UNIX (Linux, Solaris, BSD a podobne) poskytujú tieto základné bezpečnostné mechanizmy:

- Používatelia a skupiny používateľov. Každý používateľ je jednoznačne identifikovaný a môže byť vlastníkom procesov, súborov a iných objektov. Zoznam používateľov je typicky umiestnený v súbore */etc/passwd*. Používateľ môže byť zaradený ako člen skupín. Súbory, procesy a niektoré iné objekty majú priradenú jednu skupinu. Zoznam skupín a priradenie používateľov do skupín je typicky v súbore */etc/group*.

- Prístupové práva súborov. Každý súbor (alebo iný podobný objekt – súčasť súborového systému) má priradené práva na čítanie, zápis a vykonávanie pre svojho vlastníka, jednu skupinu používateľov a ostatných používateľov.

- Mechanizmus setuid. Vykonateľný súbor môže mať nastavený príznak, že bude vykonávaný s prístupovými právami jeho vlastníka. Štandardne sa súbor vykonáva s právami používateľa, ktorý ho spúšťa.

- Superužívateľ. Tento špeciálny používateľ označovaný ako root má všetky oprávnenia.

Niektoré operačné systémy poskytujú navyše ešte niektoré rozšírenia a nové bezpečnostné mechanizmy. Uvedieme príklady mechanizmov, ktoré poskytuje operačný systém Linux[2]:

- Capabilities. Každému procesu je priradená skupina oprávnení z definovanej množiny. Tieto oprávnenia sa nazývajú capabilities. Jedná sa vlastne o niektoré z práv superužívateľa.

- Access Control Lists. Štandard POSIX definuje rozhranie pre jemnejšie definovanie prístupových práv k súborom ako len vlastník, skupina a ostatní – ku každému súboru môže byť priradený zoznam používateľov s skupín spolu s príslušnými právami, ktoré k tomuto súboru majú.

- Atribúty súborov. Niektoré súborové systémy umožňujú nastaviť, aby sa súbor nedal modifikovať, alebo aby sa dalo zapisovať len na koniec súboru.

3.3 Databázový server MySQL

Databázový server MySQL je jeden z najpoužívanejších SQL serverov a je dostupný pre voľné použitie pod GPL licenciou. Tento server implementuje na riadenie prístupu k uchovávaným dátam systém privilégií. Táto časť dokumentu bližšie opisuje princípy a detaily fungovania systému privilégií MySQL servera. Predpokladá sa, že čitateľ má aspoň základné znalosti o relačných databázach

[2] od verzie 2.2

a princípoch jazyka SQL. Bližšie informácie o MySQL databáze je možné nájsť v literatúre (MYSQL).

Základnou funkciou bezpečnostného subsystému v MySQL serveri je identifikovať a autentifikovať používateľa a prideliť mu práva k operáciám so serverom, databázami, tabuľkami či stĺpcami tabuliek.

Identita používateľa je tvorená používateľským menom a menom počítača, z ktorého sa používateľ prihlasuje k MySQL serveru. Pri konfigurácii prístupových práv je možné pri špecifikovaní mena počítača používať aj skratky z jazyka SQL – napríklad znak % slúži ako skratka pre akýkoľvek reťazec znakov. Ak je ako používateľské meno udaný prázdny reťazec, vzťahuje sa táto konfigurácia na akéhokoľvek používateľa.

Riadenie prístupu v MySQL serveri sa skladá z dvoch krokov:

1. kontrola, či sa klient môže prihlásiť k serveru (autentifikácia),
2. kontrola, či má klient právo vykonať požadovanú operáciu (autorizácia).

MySQL server si informácie o prístupových právach používateľov ukladá do databázy s názvom *mysql*. Táto databáza obsahuje tabuľky *user*, *db*, *host*, *tables_priv*¸ *columns_priv*.

3.3.1 Pripojenie k serveru

Na kontrolu, či má používateľ právo pripojiť sa k serveru je používateľ identifikovaný pomocou mena, pod ktorým sa prihlasuje a adresy počítača, z ktorého sa snaží prihlásiť.

Autentifikácia sa vykonáva podľa údajov *Host*, *User* a *Password* v tabuľke *user* databázy *mysql*. V tabuľke sa vyhľadá najšpecifickejší záznam, ktorý zodpovedá identite používateľa, t.j. záznam, v ktorom sa vyskytuje najmenej skratiek. V tomto zázname sa porovná heslo, ktoré používateľ zadal s heslom v položke *Password*. Ak toto heslo súhlasí, používateľ bude pripojený k serveru. Ak sa zodpovedajúci záznam nenájde, alebo používateľ zadal nesprávne heslo, prístup bude odmietnutý.

3.3.2 Kontrola oprávnení

Keď sa používateľ úspešne autentifikuje a je pripojený k serveru, môže zadať požiadavku na vykonanie akcie. V tomto bude sa aplikuje kontrola privilégií. Pre používateľa identifikovaného svojím menom a adresou počítača, z ktorého sa pripája sa v tabuľkách databázy *mysql* server pokúsi nájsť záznamy o privilégiách vykonať požadovanú akciu. Ak tam vyhovujúce záznamy nájde, používateľ je

autorizovaný akciu vykonať. V opačnom prípade bude vykonanie akcie odmietnuté.

Tabuľka *user* prideľuje globálne privilégiá. Jednak sú to privilégiá na prácu so serverom samotným, napr. privilégium na vypnutie servera, vypísanie zoznamu databáz, vypísanie informácií o bežiacich procesoch a podobne. Okrem toho, sú tu aj globálne privilégiá na prácu s položkami, napr. privilégium na *SELECT*, *DROP* a podobne. Ak tabuľka *user* dáva používateľovi napríklad privilégium na operáciu *INSERT*, potom má používateľ právo vkladať riadky do akejkoľvek tabuľky v akejkoľvek databáze na serveri.

Tabuľky *db* a *host* slúžia na prideľovanie privilégií k jednotlivým databázam. Tabuľka *host* sa prehľadáva len vtedy, ak je v tabuľke *db* nájdený používateľ, ale prislúchajúca položka *Host* je prázdna. Tieto tabuľky poskytujú používateľovi privilégiá (typu *SELECT*, *UPDATE*, *INSERT* a pod.) pre všetky tabuľky danej databázy.

Tabuľka *tables_priv* poskytuje používateľovi privilégiá na operácie s konkrétnymi tabuľkami konkrétnej databázy.

Tabuľka *columns_priv* poskytuje privilégiá na operácie s jednotlivými stĺpcami konkrétnych tabuliek.

3.3.3 Zmeny prístupových práv

Na vykonávanie zmien prístupových práv poskytuje MySQL príkazy *GRANT* a *REVOKE*, ktoré sú súčasťou štandardov jazyka SQL. Okrem toho je možné aj priamo manipulovať s dátami v databáze *mysql* pomocou bežných príkazov na manipuláciu s dátami (*INSERT*, *UPDATE*, ...). Po vykonaní zmien v tejto databáze je potrebné vykonať príkaz *FLUSH PRIVILEGES* aby sa novo-vytvorená konfigurácia prístupových práv aplikovala.

4 Špecifikácia požiadaviek

Pri špecifikácii požiadaviek na systém ACDesigner sme vychádzali z metodológie návrhu bezpečnostných politík (časť 2.4), z praktických skúseností s návrhom politík pomocou kreslenia modelu politiky na papier. V neposlednom rade sme vychádzali aj z práce s prototypmi programu ACDesigner, ktoré vznikali v jednotlivých iteráciách vývoja.

4.1 Objekty a vzťahy medzi nimi

Model bezpečnostnej politiky v systéme ACDesigner vychádza z pohľadu na politiku ako na množinu objektov a vzťahov medzi nimi.

Ako **objekty** môžu vystupovať konkrétne komponenty bezpečnostnej politiky, napr. používatelia, súbory, privilégiá a podobne. Ako príklad **vzťahu** medzi objektami môžeme uviesť vzťah priradenia oprávnenia používateľovi alebo vzťah vzájomného vylučovania medzi rolami. Vzťahy sa chápu ako buď **vzťah medzi dvoma objektami**, v ktorom zväčša záleží na orientácii vzťahu (napríklad je rozdiel či rola A dedí od roly B alebo naopak) alebo sa môže jednať o **vzťah medzi viacerými objektami**. Vo vzťahu môžu vystupovať objekty rovnakého typu (napr. vzťah hierarchie rolí), alebo sa môže jednať o vzťah medzi objektami rôznych typov (napr. priradenie oprávnení role).

V ďalšom texte budeme používať výraz „**komponent**" pre spoločné označenie objektov a vzťahov medzi objektami.

4.2 Požiadavky na zobrazenie komponentov programu

Pre každý objekt -komponent bezpečnostnej politiky- a vzťah medzi týmito objektami je potrebné umožniť používateľovi identifikovať

1. typ komponentu – napr. používateľ, rola, priradenie oprávnenia role ...
   Typy objektov a vzťahov medzi nimi pri návrhu politík založených na RBAC budú dané komponentmi bezpečnostného modelu.

2. identitu komponentu (meno používateľa a pod.)
   Identita je dôležitá najmä pri objektoch – používateľoch, rolách a oprávneniach. Identita vzťahov medzi objektami zväčša nemá pre politiku a jej návrhára význam.

3. iné vlastnosti objektov a vzťahov medzi nimi.
   Tieto vlastnosti už nie sú súčasťou bezpečnostného modelu, ale sú významné pre politiku ako takú a pre jej implementáciu na reálny systém.
   Jednotlivé takéto vlastnosti komponentov môžu byť typu:

- text,
- číslo,
- logická hodnota (pravda/nepravda),
- hodnota z množiny (vymenovaný typ).

4. dočasné atribúty.
   Tieto atribúty objektov a vzťahov medzi nimi nie sú súčasťou navrhovanej politiky. Sú len pomocnými vlastnosťami, ktoré slúžia používateľovi programu ACDesigner.

   - objekty označené používateľom,
   - presúvané objekty,
   - objekty vyznačené systémom (napr. ako výsledok akcie používateľa).
   - skryté objekty

4.3 Požiadavky na vlastnosti vizuálneho zobrazenia ako celku

Táto časť hovorí o požiadavkách na používateľské rozhranie systému. Pri ich formulovaní sme vychádzali z (BEN02), (LR94) a (VV01), a z doterajších skúseností s tvorbou a používaním aplikácií s grafickým používateľským rozhraním.

Jednoduchosť, prehľadnosť, pochopiteľnosť pre používateľa Jedna zo základných požiadaviek na program ACDesigner je, že má zjednodušiť používateľovi pohľad na navrhovanú politiku tým, že abstrahuje od konkrétnej syntaxe konfigurácie bezpečnostných mechanizmov. Model politiky prezentovaný používateľovi by nemal byť komplikovaný, a nemal by obsahovať umelé komplikácie, ktoré do reálnej politiky nepatria.

Ak by sa používateľ v grafickom zobrazení politiky nevedel rýchlo a presne orientovať, identifikovať objekty a vzťahy medzi nimi, program by nesplnil svoju úlohu – akcelerovať návrh politiky.

Prirodzenosť a intuitívnosť používania Prostredie programu by malo simulovať prostredie známe používateľovi (napríklad papier a ceruzku). Ovládanie programu musí byť také, aby používateľ, ktorý chce vykonať nejakú akciu, mohol použiť také akcie, aké by použil aj v inom prostredí na dosiahnutie toho istého výsledku a navyše musí tieto akcie podporovať a ešte viac mu ich uľahčovať.

Akcie a vlastnosti týkajúce sa jedného objektu musia byť dostupné priamo pri tomto objekte, používateľ nesmie byť nútený presúvať svoju pozornosť inde (do hlavného menu, na panel nástrojov a pod.)

Konzistentnosť vizuálneho vyjadrenia Jedna vlastnosť musí byť znázornená vždy rovnakým spôsobom, jeden spôsob znázornenia použitý len na vyjadrenie rovnakej vlastnosti.

Neobmedzovanie používateľa Systém by nemal znemožniť používateľovi vykonať požadovanú akciu. Ak je výsledokm tejto akcie chybový stav, je na to potrebné používateľa upozorniť a umožniť mu chybu opraviť.

Ergonómia Pozornosť používateľa nesmie byť rozptyľovaná zbytočnými a pre návrh politiky nepodstatnými elementmi (zbytočnými potvrdzovacími dialógmi a pod.)

V prípade, že je však potrebné používateľa na niečo upozorniť, je potrebné čo najrýchlejšie na dané miesto používateľovu pozornosť upriamiť (napr. kontrastnou farbou, blikaním a pod.)

Vo všeobecnosti je potrebné odstupňovať kontrastnosť zobrazenia skutočností v modeli podľa ich naliehavosti a významu zobrazovanej skutočnosti pre daný moment používateľovej práce s modelom. To znamená, že pre danú činnosť nepodstatné fakty zobrazovať nekontrastným spôsobom (prípadne nezobraziť vôbec). Bežné fakty pre činnosť dôležité označovať štandardným kontrastom a výnimočné situácie (konflikty s obmedzeniami, nekonzistentnosti...) vyznačovať v danom momente vysoko kontrastne.

Estetickosť Práca s programom by nemala byť pre používateľa nepríjemná. Program je zameraný na serióznu prácu a preto by jeho vzhľad tomu mal zodpovedať decentným volením grafickej stránky zobrazenia.

Výhodou bude ak budú napríklad zladené používané farby, štýl obrázkov, tvar a typ písma a podobne.

4.4 Práca s modelom

V časti 2.4 je opísaný štandardný postup pri návrhu bezpečnostnej politiky. Z tejto teórie, ale aj zo skúseností s návrhom politík len s pomocou papiera a ceruzky i zo skúseností s testovaním prototypov programu ACDesigner sme prišli k takýmto záverom a z nich vyplývajúcim požiadavkám na program ACDesigner. V tomto opisa používame terminológiu z modelu RBAC (používatelia, role, oprávnenia, ...) ale ide o všeobecný postup, ktorý sa dá aplikovať aj pri návrhu bezpečnostnej politiky založenej na inom bezpečnostnom modeli.

Identifikovanie používateľov a oprávnení v systéme Na začiatku návrhu politiky je vhodné mať na jednej strane zoradených všetkých používateľov a na strane

druhej všetky oprávnenia. Používatelia sú pri navrhovaní politiky pre organizáciu známi a oprávnenia sú dovolené akcie v systéme, ku ktorým je potrebné riadiť prístup.

Pri návrhu na papieri sme postupovali tak, že sme na ľavej strane papiera mali pod sebou všetkých používateľov, zoskupených intuitívne, podľa ich hlavných činností v organizácii. Na pravej strane boli zasa pod sebou všetky oprávnenia v organizácii, zoradené tiež intuitívne do neformálnych skupiniek podľa ich vzťahov v organizácii.

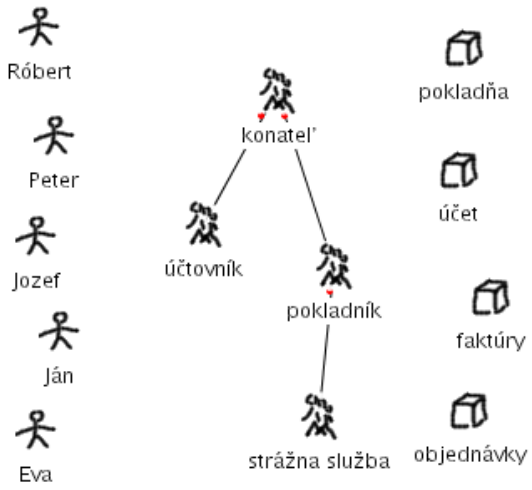
Obrázok 5: Identifikovanie používateľov a oprávnení

Úloha prostriedku ACDesigner pri podpore tejto činnosti je v automatizácii zobrazenia všetkých používateľov a oprávnení z organizácie do modelu v programe. Okrem toho program umožňuje aj voľné presúvanie, neformálne zoskupovanie objektov.

Identifikovanie rolí a ich hierarchie Analyzuje sa činnosť a organizačná štruktúra organizácie a postupne sa identifikujú roly v organizácii. Cieľom je vytvoriť stromy hierarchie rolí v organizácii.

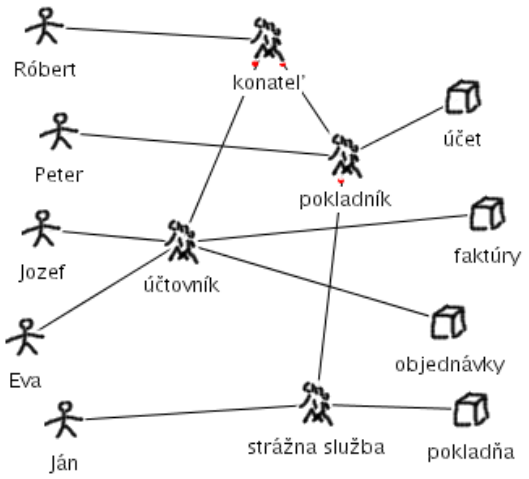
Pri návrhu politiky na papieri sme kreslili role do stredu listu medzi používateľov a oprávnenia. Hierarchiu rolí sme znázorňovali tak, že nadradené role sme kreslili nad podradené a spájali sme prislúchajúce role čiarami. Asi to nie je najvhodnejšie, ale už pri pridaní novej role sme (akosi automaticky) priraďovali do role používateľov a oprávnenia.

Program ACDesigner túto fázu návrhu politiky akceleruje tak, že umožňuje voľnosť pri pridávaní a aj odoberaní rolí, umožňuje podobným spôsobom zobraziť hierarchiu rolí.



Obrázok 6: Hierarchia rolí

Priradenie používateľov a oprávnení V tejto fáze je najdôležitejší prehľad – vedieť jednoznačne a rýchlo identifikovať objekty a vidieť čo je kam priradené.

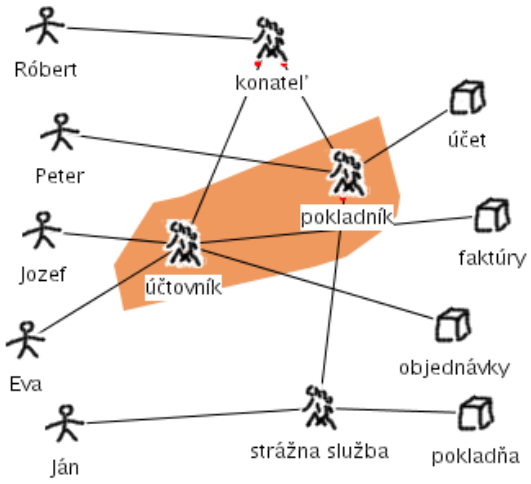


Obrázok 7: Priradenie používateľov a oprávnení

Pri práci s papierom v tejto fáze vznikali najväčšie problémy v tom, že čiary znázorňujúce priradenia sa príliš krížili, čím sa prehľadnosť výrazne znižovala.

Pomoc programu v zvýšení prehľadnosti spočíva v tom, že identita objektov politiky je výrazne označená pri každom objekte. K zamedzeniu prílišného kríženia pomáha možnosť preusporiadania umiestnenia objektov a vzťahov na ploche. Okrem toho program interaktívne na požiadanie označí všetky role používateľa, všetky oprávnenia role a podobne, čím uľahčí používateľovi orientovať sa vo vzťahoch v navrhovanej politike.

Identifikovanie obmedzení Pri práci s obmedzeniami je dôležitá najmä kontrola konzistencie vytváranej politiky s nimi a dokonca aj medzi rôznymi obmedzeniami navzájom.



Obrázok 8: Obmedzenia

Pri návrhu politiky na papieri sa kontrola konzistencie zväčša robí len nedbanlivo, odhadom. Prípadné chyby a konflikty je veľmi ťažké objaviť. Častejšie sa kontrola robí až po skončení práce na návrhu. Pri nájdení konfliktov sa návrhár vracia k predchádzajúcim etapám návrhu a opravuje ich. Tým výrazne klesá efektivita práce.

Interaktívny program ACDesigner v tejto fáze pomáha používateľovi automatickou kontrolou konzistencie obmedzení a ostatných komponentov politiky. Pri porušení nejakého obmedzenia sa konflikt okamžite kontrastne označí a používateľ vidí, v čom konflikt spočíva a môže okamžite analyzovať možnosti na jeho odstránenie.

Verifikácia navrhnutej politiky Verifikácia spočíva v kontrole, či sú dodržané princípy správne navrhnutej politiky (princíp najmenších privilégií, princíp oddelenia právomocí, ...).

Verifikácia pri ručnom návrhu politiky je zväčša taká, že sa pre jednotlivých používateľov nachádzajú všetky roly, do ktorých patrí, nájdenie všetkých oprávnení ktoré ako ich člen získa a porovnanie s jeho skutočnými potrebami na oprávnenia. Niekedy sa rovnaký postup aplikuje aj z druhej strany – vyhľadávajú sa všetci používatelia k danému oprávneniu. To znamená sledovanie množstva spojníc v zložitom modeli, ktoré jednak trvá priveľa času a jednak je náchylné na chyby.

Pomoc programu ACDesigner je v tom, že umožní na požiadanie označiť

všetky oprávnenia pre vybraného používateľa a podobne pre role a oprávnenia. To oslobodí návrhára od nutnosti vyhľadávania v spleti vzťahov a umožní mu sústrediť sa na samotnú verifikáciu správnosti navrhnutej politiky.

Verifikácia obmedzení v politike, ktoré sú zabezpečované interaktívne systémom teoreticky ani nie je potrebná, pretože ak sa model nenachádza v chybovom stave, tak sú tieto obmedzenia dodržané (za predpokladu, že systém pracuje správne).

4.5 Operácie s komponentmi

Program ACDesigner umožňuje používateľovi vykonať bežné akcie s komponentmi bezpečnostnej politiky.

- Vizuálna tvorba diagramu – manipulácia s komponentmi, pridávanie komponentov z palety, modifikácia ich rozmiestnenia v grafickom používateľskom prostredí.
- Operácie nad komponentmi:
  - mazanie,
  - editovanie parametrov (vlastností),
  - skrývanie/zobrazovanie komponentov.
- Editovanie vlastností komponentov.
- Editovanie vzťahov medzi komponentmi.
- Voliteľné zobrazovanie alebo nezobrazovanie komponentov na pracovnej ploche programu.
- Označenie skupiny komponentov a vyvolanie nejakej akcie pre celú skupinu komponentov, prípadne iba pre označené uzly, alebo hrany.
- Vyhľadanie komponentov podľa zadaných kritérií.
- Automatické rozmiestnenie komponentov na ploche.

4.6 Použiteľnosť v reálnych aplikáciách

Systém je možné priamo použiť na návrh bezpečnostnej politiky

1. založenej na RBAC bezpečnostnom modeli,
2. pre správu prístupu v operačnom systéme typu Unix,

3. pre správu prístupu v databázovom serveri MySQL.

Okrem toho systém umožňuje ukladanie dát z programu do dátového súboru (v XML formáte) a opätovné načítanie týchto dát. Umožňuje aj operácie schránky (kopírovanie a vkladanie).

4.6.1 Tvorba politiky RBAC

Systém umožňuje navrhovať všeobecnú politiku založenú na bezpečnostnom modeli RBAC. Umožňuje prácu s komponentami modelu, ich vzťahmi a obmedzeniami ako sú definované v časti 3.1.

4.6.2 Riadenie prístupu v systéme Unix

Systém ACDesigner umožňuje vytvárať politiku riadenia prístupu k objektom súborového systému Operačných systémov typu Unix. Je schopný načítať existujúce dáta o používateľoch, skupinách a súboroch v existujúcom systéme. Okrem toho je schopný načítať a interpretovať aj priradenia súborov používateľom a skupinám spolu s príslušnými prístupovými právami.

Objekty súborového systému zobrazuje v stromovej štruktúre, umožňuje operácie „rozbalenia" a „zbalenia" adresára.

Vytvorenú politiku je schopný aplikovať na skutočný systém.

4.6.3 Riadenie prístupu v MySQL databáze

Systém ACDesigner umožňuje vytvárať politiku riadenia prístupu k objektom databázového servera MySQL. Komunikuje priamo s bežiacim databázovým serverom a je z neho schopný načítať aktuálnu konfiguráciu prístupových práv a po vykonaných úpravách je schopný túto politiku aplikovať na bežiaci MySQL server.

4.7 Rozšíriteľnosť riešenia

Systém je nezávislý na použitých typoch komponentov, vstupných a výstupných dát a podobne. Programátori iných produktov môžu jednoduchým a presne definovaným spôsobom prispôsobiť ACDesigner na návrh bezpečnostnej politiky pre ich systém.

4.8 Programová manipulácia s komponentmi

Kým časť 4.5 hovorí o možnostiach, ktoré má systém ACDesigner poskytovať koncovému používateľovi, táto časť sa zaoberá požiadavkami na možnosti manipulácie s komponentmi, ktoré bude vykonávať samotný systém. Tieto možnosti sú

prostredníctvom definovaného rozhrania poskytnuté programátorom iných produktov, ktorí budú rozširovať ACDesigner na návrh politiky pre ich systém.

- Definovanie typov komponentov. Každý systém má špecifickú množinu objektov bezpečnostnej politiky a možných vzťahov medzi týmito objektami. Programátor má možnosť definovať tieto typy, spolu s ich vlastnosťami.
- Definovanie operácií nad komponentami. Programátor má možnosť definovať operácie, ktoré systém umožní používateľovi vykonávať. Jedná sa najmä o operácie s komponentami bezpečnostnej politiky, ale aj o globálne chápané operácie nad celou vytváranou politikou.
- Definovanie automatických akcií spúšťaných pri nejakej udalosti. Ak používateľ vykoná nejakú akciu, systém umožňuje automaticky reagovať na túto akciu vykonaním nejakej definovanej činnosti. Systém tiež umožní zabrániť vykonaniu akcie, v prípade, že výsledok akcie odporuje nejakému stanovenému pravidlu. Udalosťami v programe sú najmä:
  - pridanie alebo zrušenie komponentu bezpečnostnej politiky,
  - vytvorenie alebo zrušenie vzťahu medzi komponentami,
  - zmena vlastnosti komponentu alebo vzťahu medzi komponentami

5 Analýza požiadaviek

5.1 Zobrazenie komponentov politiky do dvojrozmerného priestoru

Pretože úlohou vytváraného systému je vizuálne navrhovať bezpečnostnú politiku, je potrebné zobraziť komponenty bezpečnostnej politiky a vlastnosti medzi nimi do priestoru, v ktorom s nimi bude používateľ manipulovať.

Ľudia sú navyknutí vyjadrovať myšlienky v dvojrozmernom zápise – pomocou písma, nákresov, ale aj počítačovej obrazovky. Okrem toho možnosti súčasných bežných pracovných staníc nie sú také, aby bolo bežné na nich pracovať v trojrozmernom priestore. Preto sme sa rozhodli, že práca bude prebiehať na dvojrozmernej ploche.

V dvojrozmernom priestore je možné zobraziť objekty:

- body – bezrozmerné objekty. Jeden bod na dvojrozmernej ploche je možné označiť napríklad malým obrázkom (ikonou).
- čiary – jednorozmerné objekty. Na zobrazenie jednorozmerných objektov je možné použiť čiary rôznych vlastností.
- oblasti – dvojrozmerné objekty. Oblasti predstavujú časti celej plochy a môžu byť vyznačené napríklad rôznou farbou oblasti.

Ostatné vyjadrovacie možnosti dvojrozmernej plochy zahŕňajú:

- rozmery objektu – veľkosť ikony, hrúbka čiary a podobne,
- farba objektu, jej sýtosť, jas, odtieň, kombinácia farieb pre jeden komponent,
- rôzne pozadie objektov – farba, orámovanie a podobne,
- textové označenia objektov – opisy pri objektoch s rôznou veľkosťou, typom, farbou písma.

5.2 Možnosti vizuálneho zobrazenia dvojrozmerných objektov

Ako možné komponenty v dvojrozmernom priestore sme identifikovali body, čiary a oblasti. V tejto časti analyzujeme vhodnosť jednotlivých spôsobov ich zobrazenia pre účely vizuálneho stvárnenia návrhu bezpečnostnej politiky.

Vychádzame zo zaužívaných štandardov návrhu používateľského rozhrania, najmä (BEN02) a z literatúry zaoberajúcej sa použiteľnosťou aplikácií a rozhraním medzi človekom a počítačom (LR94), (VV01).

5.2.1 Možnosti zobrazenia bodov

Obrázok – ikona Štandardy určujú použitie ikony zväčša na rozlíšenie typu alebo identity daného objektu.

Veľkosť ikony Veľkosť ikony by mala nejakým spôsobom odrážať nejakú kvantitatívnu alebo kvalitatívnu vlastnosť objektu, ktorý zobrazuje. Z hľadiska ergonómie a estetiky však nie sú vhodné prílišné rozdiely veľkosti jednotlivých ikon na ploche.

Farba ikony Farba ikony môže znázorňovať mnoho rôznych vlastností objektu. Obmedzenia pri používaní farieb sú najmä estetické – nie je pekné miešanie objektov s výrazne neladiacimi farbami. Okrem toho, človek vie naraz sledovať a bez problémov rozpoznávať len obmedzený počet rôznych farieb.

Sýtosť (plná/sivá) Zasivené ikony sa štandardne používajú na označenie neaktívnych objektov. Interpretácia neaktívnych objektov v bezpečnostnej politike môže byť rôzna. Napríklad takto môžu byť označené nepriradené uzly, uzly nesúvisiace s práve spracovávanou oblasťou diagramu a podobne.

Okrem zníženia kontrastu ikony zasivením jej farby je možné kontrastnosť aj zvýšiť – zvýšením jasu alebo saturácie farby. Takéto ikony so zvýšeným kontrastom môžu byť použité na zobrazenie dočasných atribútov, napr. označenie objektov, o ktoré používateľ požiadal, zobrazenie konfliktov s obmedzeniami a pod.

Orámovanie čiarou Existencia orámovania objektu sa v štandardoch používa na zobrazenie objektov, ktoré používateľ označil a chystá sa nad nimi vykonať nejakú operáciu.

Tvar orámovania – pravouhlý, kružnica, ... Tvar orámovania môže napríklad rozlišovať objekty označené používateľom a objekty z nejakého dôvodu označené systémom.

Pozadie (orámovanie plochou) – farba pozadia Orámovanie plochou sa v niektorých programoch používa na zobrazenie označených objektov. Vtedy sa však zvyčajne označenému objektu modifikuje farba popredia (napríklad na inverznú), čo podľa nášho názoru mätie používateľa.

Použitie farebnej výplne jednotlivého uzla neodporúčame použiť aj s ohľadom na to, že pomocou farebného pozadia je veľmi vhodné označovať oblasti.

Text pri ikone Text označuje identitu alebo iné textové vlastnosti objektu.

Značka pri ikone (hviedzička a pod.) Drobné značky pri ikonách môžu svojou existenciou alebo neexistenciou označovať hodnotu nejakej vlastnosti typu logická hodnota.
Množstvo takýchto značiek pri jednej ikone však môže znížiť prehľadnosť. Rovnako môže malá veľkosť značky sťažovať rozlíšenie jej tvaru a odlíšenie od iných značiek.

Výrazné označenie ikony – preškrtnutie, podčiarknutie, ... Výrazné označenie je síce kontrastné, avšak nie je pre používateľa nepríjemné aj keby bolo aplikované dlhšiu dobu. Takýmto spôsobom môžu byť napríklad trvalo označené objekty, ktoré sú nejakým spôsobom výnimočné (napr. nejaká ich vlastnosť alebo vzťah je v konflikte s obmedzeniami) a nie je potrebné na ne okamžite upútať pozornosť používateľa (napríklad preto, že na ne už raz upozornený bol).

5.2.2 Možnosti zobrazenia čiar

Farba Na farbu čiar sa vzťahujú rovnaké možnosti a obmedzenia ako na farbu ikon.

Hrúbka Hrúbka čiary v existujúcich systémoch zväčša znázorňuje nejakú číselnú vlastnosť vzťahu. Z estetického hľadiska nie je vhodné príliš veľké rozpätie hrúbok čiar.

Typ (plná, prerušovaná, ...) Typom čiary je veľmi vhodné označovať hodnotu nejakej vlastnosti vymenovaného typu.

Tvar (rovná, vlnovková, ...) Podobne ako typ čiary. Kombinácie rôznych typov a tvarov čiar nemusia v podstate spôsobovať problémy, je však potrebné si na to dať pozor, najmä keď by týchto kombinácií bolo veľa.

Text pri čiare Text môže (podobne ako pri ikone) označovať typ vzťahu, alebo identitu vzťahu (tá však zväčša nemá význam) alebo hodnotu nejakej inej textovej vlastnosti.

Značky pri čiare (hviezdička a pod.) Značky pri čiare majú podobné vyjadrovacie schopnosti ako značky pri ikone.

Tvar koncov čiary (šípky, ...) Šípky je veľmi vhodné použiť na označenie orientácie vzťahu medzi dvoma objektami.

Zvýraznenie začiatku a konca čiary Takto sa štandardne označujú čiary označené používateľom.

5.2.3 Možnosti zobrazenia oblastí

Farba pozadia Najjednoduchšie a najprirodzenejšie zobrazenie oblasti je jej vyplnenie farbou. V tomto prípade môžu nastať problémy, ak sa oblasti dotýkajú alebo dokonca prekrývajú.

Orámovanie hraníc oblasti Pri veľkých oblastiach a veľkom množstve oblastí môže používateľ ťažšie rozlíšiť vnútro a vonkajšok takto znázornených oblastí.

Šrafovanie oblastí Vyplnenie oblastí rôznymi druhmi šrafovania môže slúžiť aj na označenie oblasti ako takej, ale aj na označenie nejakej vlastnosti oblasti.

5.2.4 Kombinácie zobrazení

Už pri opise obmedzení pri jednotlivých možnostiach zobrazenia komponentov sme narazili na niektoré možné problémy, ktoré by mohli vzniknúť pri nevhodnom kombinovaní spôsobov zobrazenia.

Veľký počet rôznych komponentov a znázornení na pracovnej ploche môže miasť používateľa a namiesto výhody sa stane vizuálnosť zobrazenia nevýhodou. Obmedzovať počet komponentov, ktorý si na plochu používateľ umiestni nie je možné, takže je potrebné dbať na striedmosť pri počte rôznych typov zobrazenia komponentov, ich vzťahov a vlastností.

Okrem počtu a rozmanitosti komponentov môže spôsobovať problémy aj rovnaké alebo príliš podobné vizuálne vnímanie rôznych zobrazovaných skutočností. Nesmie sa napríklad stať, aby objekt farebne splýval s pozadím, aby sa dva objekty prekrývali tak aby niektorý z nich nebolo vidieť a podobne.

5.3 Analýza možných implementačných prostriedkov

Ako hlavné požiadavky ovplyvňujúce výber implementačného prostredia pre program ACDesigner sa ukázali byť

- požiadavky na implementáciu grafického prostredia v dvojrozmernom priestore,
- univerzálnosť v zmysle prenositeľnosti na rôzne platformy,
- modulárnosť – možnosť jednoduchej implementácie modulov programu, použiteľných za behu a možnosť vytvoriť čo najpoužiteľnejšie rozhranie pre programátorov modulov,

Ako možné platformy boli zvažované nasledovné:

GTK – knižnica v jazyku C pre aplikácie v grafickom používateľskom rozhraní. Knižnica je použiteľná na unixových operačných systémoch, s podporou knižnice cygwin aj na platformách typu Windows. Modulárnosť je vzhľadom na použitie jazyka C obmedzená. (GTK)

QT je GUI knižnica pre jazyk C++. Programy napísané s jej použitím je možné kompilovať na unixovýh platformách (nekomerčné projekty zadarmo) aj na platforme Windows (s komerčnou verziou knižnice). Modulárnosť je podporovaná priamo knižnicou, implementácia rozhraní na moduly však nie je ideálna. (QT)

TCL je skriptovací jazyk a možnosťami použitia pre aplikácie v grafickom používateľskom rozhraní. Programy sú platformovo nezávislé, pokiaľ na konkrétnej platforme existuje interpreter jazyka tcl. Obmedzujúcim faktorom tohto riešenia je nevyhnutnosť inštalácie externých komponentov (knižníc v tcl), pretože grafické možnosti interpretra sú bez nich nedostatočné. (TCL)

Visual Basic je proprietárne riešenie firmy Microsoft na rýchly vývoj aplikácií v grafickom používateľskom rozhraní. Jeho výhodou je veľmi dobrá prepracovanosť grafických komponentov a štandardizovaný spôsob rozširovateľnosti cez technológiu COM, DCOM, príp. .NET. Nevýhodou je obmedzenosť na platformu operačných systémov Windows. (VB)

Delphi/Kilyx sú prostredia vyvinuté firmou Borland. Ako programovací jazyk sa používa Pascal, grafické možnosti sú na vysokej úrovni. Výhodou by bolo, keby programy v zdrojovom tvare boli úplne prenositeľné medzi produktami Delphi a Kylix. Tým by sa stali prenositeľnými na platformách Windows a Linux. (DEL)

Java je multiplatformové, voľne šíriteľné riešenie vyvinuté firmou Sun. So štandardným balíkom Swing má aj veľmi dobré grafické vlastnosti. Programy sú prenositeľné medzi platformami aj v skompilovanom tvare, pretože sa vykonávajú interpretovane. Celý jazyk je silne objektovo orientovaný. Dynamické zavádzanie modulov do programu je triviálne, stačí naprogramovať triedu, skompilovať ju a program môže začať vytvárať inštancie tejto skompilovanej triedy. (JAVA)

6 Návrh riešenia

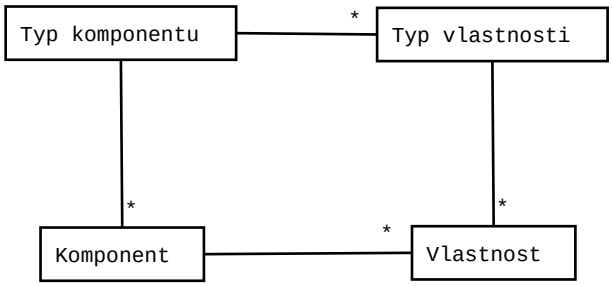
6.1 Diagram bezpečnostnej politiky

Modelom navrhovanej politiky používaným v programe Access Control Designer je **diagram bezpečnostnej politiky**. Na obrázku 9 je znázornený konceptuálny dátový model systému. Komponenty diagramu bezpečnostnej politiky sú:

- **uzly** – jednotlivé entity bezpečnostnej politiky – používatelia, oprávnenia a podobne.
- **hrany** – orientované spojnice medzi dvoma uzlami. Označujú nejaký vzťah medzi spojenými entitami.
- **oblasti** – množiny uzlov. Príslušnosť uzlov k nejakej triede uzlov.

Ku každému komponentu sú navyše priradené

- **vlastnosti** – dáta rôznych typov a významu
- **metódy** – funkcie, ktoré sa týkajú daného komponentu a sú spúšťané na požiadavku používateľa
- **udalosti** – podobne ako metódy, funkcie obsluhy udalostí sú funkcie priradené komponentom, avšak spúšťajú sa automaticky po nejakej akcii súvisiacej s daným komponentom



Obrázok 9: Logický model dát v systéme

6.2 Interaktívny diagram objektov a vzťahov

Pri návrhu programu ACDesigner vznikol koncept **interaktívneho diagramu**, ktorý je možné použiť ako univerzálny komponent aj pre iné programy, ktoré majú za úlohu modelovať alebo navrhovať systémy, v ktorých sú dôležité objekty a vzťahy

medzi nimi. Základným konceptom interaktívneho diagramu je koncept uzlov, hrán, oblastí a k nim prislúchajúcich vlastností, metód a udalostí.

Modulárnosť riešenia a objektovosť návrhu a implementácie umožňuje znovupoužitie interaktívneho diagramu ako komponentu v iných programoch.

6.3 Reprezentácia komponentov vybraných bezpečnostných modelov

V tejto časti je opísaná navrhovaná reprezentácia objektov a vzťahov v bezpečnostných politikách, implementovaných v operačných systémoch typu Unix, v databázovom systéme MySQL a vo všeobecnom bezpečnostnom modeli RBAC.

6.3.1 Operačný systém Unix

Používateľ je reprezentovaný uzlom diagramu. Jeho vlastnosťami sú prihlasovacie meno, heslo (v zašifrovanom tvare), uid-číslo, reálne meno, shell a domovský adresár.

Skupina používateľov je tiež reprezentovaná uzlom. Jej vlastnosťami sú meno skupiny, prístupové heslo a gid-číslo.

Medzi používateľom a skupinou sú definované dva vzťahy – vzťah „login group" je reprezentovaný hranou medzi používateľom a skupinou. Vzťah „group" je tiež reprezentovaný hranou medzi uzlom používateľa a skupiny. Na tieto typy hrán sa vzťahujú nasledovné obmedzenia:

- Každý používateľ má práve jednu skupinu, s ktorou má hranu „login group". Ak práve nemá priradenú žiadnu skupinu ako svoju „login group", musí byť kontrastne označený – v chybovom stave.
- Ak sa používateľ systému ACDesigner pokúsi priradiť používateľovi ďalšiu skupinu, táto skupina nebude priradená ako „login group" ale ako „group".
- Používateľ systému ACDesigner môže zmeniť hranu „group" na „login group". V takom prípade sa pôvodná hrana „login group" zmení na „group".

Súbor je reprezentovaný uzlom a jeho vlastnosti sú meno súboru, právo ostatných na čítanie, zapisovanie a vykonávanie. Adresár je rovnako reprezentovaný uzlom a jeho vlastnosťami sú meno adresára a prístupové práva ostatných na prezeranie, zapisovanie a vstúpenie do adresára. Medzi súborom a adresárom je vzťah „je v adresári" reprezentovaný hranou. Rovnako vzťah medzi dvoma adresármi „je podadresár" je reprezentovaný hranou.

Vzťahy medzi používateľom resp. skupinou a súborom resp. adresárov sú reprezentované hranami. Tieto hrany majú ako vlastnosti prístupové práva na

čítanie, zapisovanie a vykonávanie, resp. vstúpenie do adresára pre vlastníka alebo skupinu.

6.3.2 Databázový server MySQL

Reprezentáciou používateľa systému MySQL je uzol s vlastnosťami meno používateľa, počítač, z ktorého sa prihlasuje a prístupové heslo.

Databáza, tabuľka a stĺpec tabuľky sú reprezentované uzlami, ktoré majú ako vlastnosť uvedené meno príslušného objektu. Vzťahy tabuľka patrí do databázy a stĺpec patrí do tabuľky sú reprezentované hranami.

Uzlom je reprezentované aj oprávnenie. Ako vlastnosť oprávnenia je uvedený názov oprávnenia, ktorý sa zhoduje s názvom operácie, ktorú toto oprávnenie umožňuje (napr. Select, Update a pod.)

Vzťah, že oprávnenie je priradené používateľovi je reprezentovaný hranou medzi používateľom a oprávnením.

To, k akému objektu sa oprávnenie vzťahuje je reprezentované hranou medzi oprávnením a týmto objektom. To znamená, že napríklad oprávnenie vykonať operáciu Select nad tabuľkou Tbl je vyjadrené uzlom typu oprávnenie s menom Select spojeným hranou s uzlom typu tabuľka s názvom Tbl.

Serverové oprávnenia (super-právomoci) sú vyjadrené uzlom oprávnenia, ktorý nie je spojený so žiadnou databázou, tabuľkou, či stĺpcom.

6.3.3 Bezpečnostný model RBAC

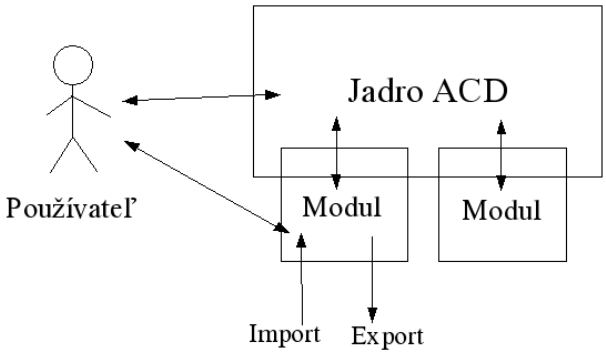
Vo všeobecnom modeli RBAC politiky je používateľ reprezentovaný uzlom diagramu a jeho vlastnosťou je jeho meno. Rola je reprezentovaná taktiež hranou, ktorej vlastnosťou je názov role. Rovnako aj oprávnenie je reprezentované uzlom s menom.

Vzťah príslušnosti používateľa k role je reprezentovaný hranou medzi uzlom používateľa a role. Vzťah priradenia oprávnenia role je rovnako reprezentovaný hranou medzi rolou a oprávnením.

Hierarchia rolí je reprezentovaná hranou medzi nadradenou a podradenou rolou v hierarchii. Vzťah vzájomného vylučovania medzi rolami je reprezentovaný oblasťou, do ktorej môžu patriť role. V prípade, že sa dve role nachádzajú v jednej oblasti, príslušnosť do týchto rolí sa vzájomne vylučuje.

6.4 Architektúra programu

Architektúra programu Access Control Designer je modulárna. Na obrázku 10 je schematicky znázornená funkcia jadra programu ACDesigner a modulov programu.



Obrázok 10: Architektúra programu

6.4.1 Jadro programu

Jadro je nezávislé na type navrhovanej bezpečnostnej politiky a len vykonáva samotné manipulácie s komponentmi, vykonáva metódy komponentov a stará sa o vyvolávanie a spracovanie udalostí.

Toto jadro je navrhnuté ako nezávislý znovupoužiteľný komponent, ktorý je možné použiť aj v iných nástrojoch na prácu s interaktívnymi diagramami.

6.4.2 Zásuvné moduly programu

Rozhranie programu na moduly umožňuje aj externým autorom vytvárať moduly, pomocou ktorých bude možné program používať na návrh bezpečnostných politík v systémoch s rôznymi bezpečnostnými mechanizmami, typmi komponentov bezpečnostnej politiky a podobne. Moduly do systému registrujú:

- **typy komponentov** – napr. RBAC modul zaregistruje komponenty používateľ, oprávnenie, rolu, priradenie používateľov do rolí , priradenie oprávnenia role atď.
- **typy vlastností/parametrov komponentov** – typom vlastnosti môže byť napríklad meno používateľa, prístupový mód pri priradení oprávnenia a podobe.
- **obsluha udalostí** – metódy komponentov, ktoré sa aktivujú automaticky pri nejakej akcii týkajúcej sa komponentu – udalosti, napríklad udalosť, ktorá sa aktivuje pri pridaní používateľa do role a pod. Udalosti môžu napríklad upozorniť používateľa na porušenie obmedzení pri aplikovaní požadovanej akcie, alebo zabezpečia automatické vykonanie nejakej inej akcie ako následku používateľovej akcie.

- **globálne funkcie** – môžu slúžiť napríklad na import a export dát z vonkajšieho prostredia alebo na akcie, ktoré nesúvisia s jedným komponentom ale s celým diagramom.

6.4.3 Rozhrania

Program poskytuje dve rozhrania:

1. používateľské rozhranie – prostredníctvom neho komunikuje s používateľom,
2. rozhranie na zásuvné moduly – na komunikáciu s modulmi programu.

6.5 Práca s vstupnými a výstupnými dátami

Vytvorený diagram bezpečnostnej politiky sa transformuje na pravidlá riadenia prístupu, ktoré budú implementované mechanizmami príslušného systému.

Pomocou zásuvných modulov je realizovaný vstup dát z vonkajšieho prostredia a výstup navrhnutej politiky vo formáte určenom pre dané bezpečnostné mechanizmy. Táto výmena údajov nástroja a okolitého sveta je do používateľského rozhrania zahrnutá ako voľby importu a exportu dát medzi globálnymi funkciami programu. Funkcie zabezpečujúce import a export údajov môžu buď komunikovať priamo so systémom, ktorého bezpečnostná politika sa navrhuje (napr. v reálnom čase zadávať SQL príkazy do databázy a pod.) alebo môže len načítavať a spätne zapísať konfiguračný súbor pre systém a ten si ich potom musí prečítať.

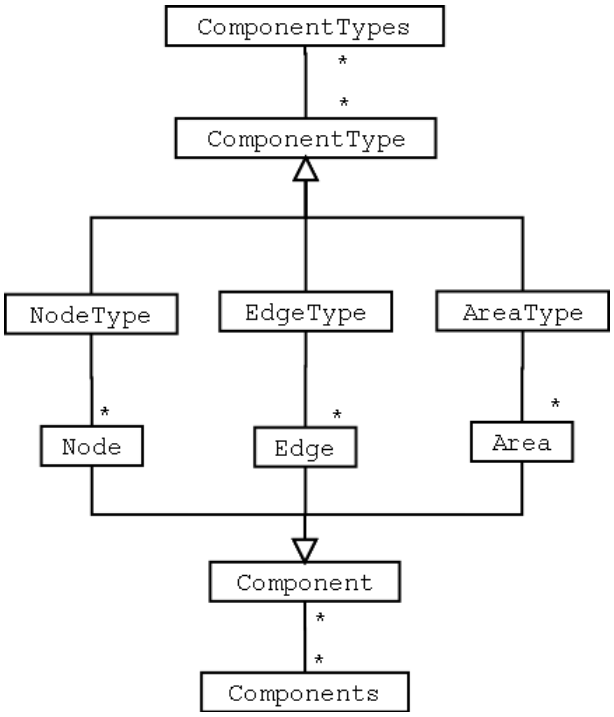
Jeden z modulov dáta ukladá do univerzálneho formátu založenom na formáte XML (BPS00) a tieto uložené dáta potom aj načíta. Tento typ importno-exportného filtra plní funkciu klasického ukladania a otvárania súborov aplikácie. To znamená, že z dát v tomto súbore bude možné úplne obnoviť stav programu, ktorý bol v momente uloženia dát do súboru. Okrem toho, tento modul implementuje aj funkcionalitu kopírovania a vloženia (schránky).

6.6 Model dát systému

Táto časť dokumentu obsahuje návrh dátového modelu systému. Rovnako ako celý návrh, aj návrh dátového modelu vychádza z objektovo-orientovanej paradigmy. Diagramy sú zapísané syntaxou jazyka UML. Názvy tried sú uvádzané v angličtine, tak ako sú pomenované v zdrojovom kóde programu. Ku každej triede je stručne uvedený jej opis a význam pre program, sú uvedené aj jej základné atribúty, prípadne aj metódy.

6.6.1 Komponenty a typy komponentov

Na obrázku 11 je znázornená časť dátového modelu aplikácie týkajúca sa komponentov a ich typov. Entita typ komponentu (ComponentType) je generalizáciou entít typ uzla (NodeType), typ hrany (EdgeType) a typ oblasti (AreaType). Atribútmi typu komponentu sú názov typu, informácia o tom, či majú byť objekty tohto typu skryté alebo zobrazené. Typ uzla má navyše priradenú ikonu, ktorou budú tieto komponenty reprezentované. Typ hrany má vlastnosti farbu, typ čiary. Atribútom typu oblasti je farba.



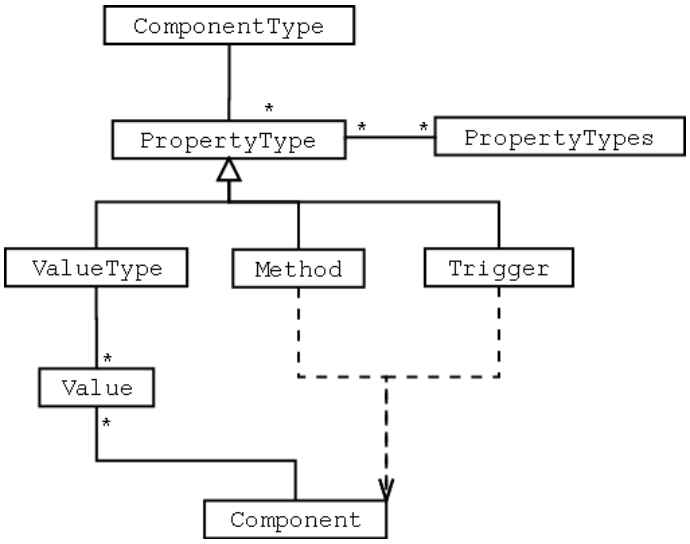
Obrázok 11: Komponenty a typy komponentov

Entita komponent (Component) je generalizáciou entít uzol (Node), hrana (Edge) a oblasť (Area). Atribútom komponentu je informácia, či je komponent skrytý, či je označený. Uzol má navyše priradené svoje súradnice, hrana zase referenciu na začiatočný a koncový uzol. Atribútmi oblasti sú súradnice hraničných bodov oblasti.

Entita typy komponentov (ComponentTypes) je kontajnerom na ukladanie objektov typov komponentov, obdobne entita komponenty (Components) pre ukladanie objektov komponentov.

6.6.2 Vlastnosti, metódy a udalosti

Na obrázku 12 je časť dátového modelu aplikácie týkajúca sa vlastností, metód a obsluhy udalostí. Typ hodnoty (ValueType), metóda (Method) a funkcia spra-



Obrázok 12: Vlastnosti, metódy a obsluha udalostí

covania udalostí (Trigger) sú zovšeobecnené ako typy vlastností (PropertyType). Typy vlastností sú priradené typu komponentu. Atribútom typu komponentu je meno typu. Typ hodnoty má navyše priradený typ – text, číslo, hodnota z vymenovaného typu a pod.

Metódy a funkcie obsluhy udalostí nie sú priradené jednotlivým komponentom, ale samozrejme vykonávať sa budú nad konkrétnymi komponentmi. Konkrétny komponent je argumentom metódy alebo funkcie obsluhy udalostí. Hodnota (Value) má priradený typ hodnoty a je priradená konkrétnemu komponentu.

Objekty – typy vlastností sa budú ukladať do kontajnerov entity typy vlastností (Property Types).

7 Implementácia softvérového produktu

Táto časť dokumentu obsahuje opis implementácie systému ACDesigner. Okrem tohto, pomerne málo štrukturovaného textu, dokumentuje implementáciu aj vysoko štrukturovaná dokumentácia k zdrojovému kódu – javadoc-dokumentácia, ktorú je možné nájsť v elektronickej forme na priloženom elektronickom médiu.

Okrem toho, v časti 10 je zvláštna príručka pre autorov zásuvných modulov systému ACDesigner.

7.1 Implementačné prostredie

Ako platforma pre vývoj programu Access Control Designer bol vybraný jazyk **Java**. Jeho hlavné výhody, vplývajúce na rozhodnutie boli najmä:

- prenositeľnosť – programy a moduly v Jave sú na úrovni bajtového kódu prenositeľné na akúkoľvek platformu, pre ktorú existuje prostredie pre beh v reálnom čase.
- grafické možnosti, reprezentované knižnicou komponentov *Swing*, ktorá je štandardnou súčasťou súčasných verzií prostredia Java.
- možnosť nahrávania modulov za behu programu. Kód jednotlivých tried sa nahráva za behu aj pri ich štandardnom použití, ide o principiálne riešenie, ktoré sa dá veľmi výhodne použiť pri modulárnej architektúre programu.
- dokumentovateľnosť – pomocou prostriedku *javadoc* je možné priamo v súbore so zdrojovým kódom dokumentovať jeho použitie. Tento prostriedok vie zo zdrojového kódu a príslušných formátovaných poznámok vytvoriť dokumentáciu, ktorá môže slúžiť ako dokumentácia k implementácii a aj ako API dokumentácia pre programátorov zásuvných modulov.

Ako nevýhody Javy je možné uviesť vysoké hardvérové nároky (najmä pri použití komponentov z knižnice Swing) a nižšia efektivita z hľadiska využitia procesora a pamäti. Tieto obmedzenia však vzhľadom na možnosti súčasných bežných pracovných staníc môžeme prijať.

7.2 Základné triedy programu

Táto časť dokumentácie je voľným opisom implementácie jednotlivých tried programu. Spolu s dokumentáciou k zdrojového kódu (javadoc-dokumentáciou) môže slúžiť ako doplnok programátorskej príručky (časť 10).

7.2.1 Komponenty

Základná trieda komponentu `Component` obsahuje premenné a funkcionalitu spoločnú pre všetky komponenty – premenná `type` je referenciou na typ komponentu, premenná `values` je odkazom na kontajner vlastností komponentu. Celočíselná premenná `id` je jednoznačný identifikátor komponentu. Trieda `Component` obsahuje aj premenné, ktoré hovoria o aktuálnom stave komponentu – či je komponent označený používateľom, alebo je skrytý a podobne. Každý komponent implementuje aj metódu `isHit`, ktorá pre zadané súradnice plochy zistí, či tieto súradnice zasahujú komponent. To je využité pri identifikovaní komponentu pri akciách používateľa. Komponenty ďalej implementujú aj metódu `draw`, ktorá slúži na vykreslenie daného komponentu.

Vlastnosti (`Value`) sa komponentu priradia okamžite po vytvorení komponentu, teda nie je potrebné počas behu programu pridávať vlastnosti pomocou metódy `addValue`. Na prístup k jednotlivým vlastnostiam komponentov implementuje trieda `Component` metódy, ktoré vyhľadajú vlastnosť podľa jej mena alebo identifikátora.

Trieda `Node` je implementáciou uzla interaktívneho diagramu. Uzol má uložené súradnice (premenné `xCoord` a `yCoord`), udržuje si zoznam všetkých hrán, ktoré sú s ním združené (premenné `edgesFrom` a `edgesTo`). Okrem toho implementuje aj metódy na vyhľadanie všetkých uzlov, s ktorými je uzol spojený hranou príslušného typu (`nodesConnectedToIterator` a `nodesConnectedFromIterator`).

Trieda `Edge` implementuje hranu interaktívneho diagramu. Okrem referencií na začiatočný a koncový uzol si hrana uchováva aj svoju farbu. Na vykresľovanie hrany a na implementáciu metódy `isHit` je použitá vlastná implementácia reprezentácie čiary, pretože nám nevyhovovala implementácia v knižnici awt javy.

Trieda `Area` implementuje oblasť. Oblasť je reprezentovaná polygónom, konkrétne objektom triedy `java.awt.Polygon`. Metódy na pridávanie a zmeny polohy hraničného bodu polygónu implementujú jednoduchý algoritmus na elimináciu nepotrebných bodov. Bod sa považuje za nepotrebný, ak leží blízko iného bodu, alebo ak leží na spojnici svojich susedných bodov. Keďže manipulácia s oblasťou je vlastne manipuláciu s hraničnými bodmi polygónu, implementuje `Area` okrem metódy `isHit` aj metódu `getPointHit`, ktorá vyhľadá hraničný bod polygónu na daných súradniciach. `Area` implementuje aj metódu `getNodes`, ktorá vyhľadá všetky uzly, ktoré sa v oblasti nachádzajú.

7.2.2 Typy komponentov

Trieda `ComponentType` implementuje spoločné vlastnosti a metódy pre všetky typy komponentov. Obsahuje meno typu (premenná `name`), kontajnery na typy vlast-

ností, metód a funkcií obsluhy udalostí. Okrem toho obsahuje aj jednoznačný identifikátor typu komponentu a informáciu o tom, či majú byť všetky komponenty tohto typu skryté alebo nie.

Metódy `addValueType`, `addMethod`, `addTrigger` na pridávanie typov vlastností, metód a funkcií obsluhy udalostí sú určené pre autorov zásuvných modulov.

V triede `ComponentType` je implementovaná aj vnútorná trieda `ACDRgb`, ktorá uľahčuje prácu s farbami.

Pri konštruovaní objektu typu `NodeType` je potrebné špecifikovať ikonu, ktorou budú uzly vytváraného typu reprezentované. Trieda `NodeType` potom obsahuje metódy na manipuláciu s touto ikonou – vytvorenie zasivenej, zafarbenej verzie ikony atď.

V triede `EdgeType` sú ako statické metódy implementované niektoré štandardné metódy na obsluhu udalostí – metóda na zamedzenie vytvorenia dvoch hrán rovnakého typu medzi dvoma uzlami (`denyDoubleEdges`), metódy, ktoré umožnia len jednu hranu daného typu vedúcu z resp. do uzla (`replaceDuplicateTo`, `replaceDuplicateFrom`). Ak sa používateľ pokúsi vytvoriť novú hranu s iným uzlom, stará hrana bude zrušená.

7.2.3 Vlastnosti, metódy a obsluha udalostí komponentov

Podľa logického dátového modelu aplikácie (časť 6.6.2) sú typy vlastností komponentov, metódy, ktoré môže používateľ vykonávať nad komponentmi a metódy obsluhy udalostí nad komponentami implementované ako triedy. Spoločnou nadradenou triedou týchto tried je trieda `PropertyType`. Tá obsahuje meno vlastnosti a jednoznačný identifikátor. Okrem toho obsahuje referenciu na typ komponentu, ku ktorému sa vzťahuje.

Metóda komponentu je trieda `Method`, ktorá obsahuje funkciu `body`, ktorá sa vykoná, keď používateľ požiada o vykonanie danej metódy nad komponentom. Ako argument táto metódy dostane konkrétny komponent, nad ktorým sa má operácia vykonať.

Trieda `Trigger` implementuje obslužnú funkciu udalosti. Podľa mena vytvoreného objektu sa rozlišujú 2 druhy týchto udalostí a k nim prislúchajúce obslužné funkcie:

1. udalosť pridania komponentu. Objekt triedy Trigger musí mať meno „`add`“.
2. udalosť odobrania komponentu. Objekt triedy Trigger musí mať meno „`remove`“.

Samotná funkcionalita, ktorá sa má vykonať pri danej udalosti nad objektom je obdobne ako v triede `Method` implementovaná v metóde `body`. Táto dostane ako argument konkrétny objekt, ktorého sa udalosť týka. Obslužná funkcia

môže zabrániť vykonaniu udalosti a to tak, že metóda body vráti hodnotu `false`. Napríklad, ak používateľ požiadal o zmazanie komponentu, vykoná sa nad týmto komponentom metóda obsluhy udalosti s názvom „`remove`" a ak jej funkcia body vráti hodnotu `true`, bude komponent zmazaný. V opačnom prípade zmazaný nebude.

7.2.4 Trieda aplikácie

Základná funkčnosť aplikácie je implementovaná v triede `ACD`. Objekt tejto triedy sa vytvorí pri spustení aplikácie a v tomto objekte sa udržiavajú zoznamy všetkých komponentov, typov komponentov, vlastností a ich typov. Okrem toho, vytvorí sa tu aj objekt triedy `Gui`, ktorý implementuje funkcionalitu používateľského rozhrania, vytvoria sa tu aj základné globálne funkcie programu (napr. zmazanie komponentov modelu, ukončenie programu. Tieto sa pridajú do hlavného menu programu.

Trieda `ACD` je zodpovedná aj za nahranie a inicializovanie modulov programu – na to slúži metóda `loadModule`. V triede `ACD` sú aj metódy na pridávanie a odoberanie komponentov (napr. `addNode`, `addEdge`...). Tieto metódy sú zodpovedné za spúšťanie príslušných funkcií obsluhy udalostí nad danými komponentmi.

Ďalej trieda obsahuje metódy na vyhľadávanie komponentov a ich typov – napríklad metódy `getNodesIterator`, `getEdgeTypesIterator`.

7.2.5 Používateľské rozhranie

Trieda `Gui` implementuje rozhranie pre používateľa. Vytvára sa tu hlavné okno aplikácie, spolu so všetkými svojimi komponentami a implementuje sa funkčnosť týchto komponentov.

Vnútorná trieda `Gui.ACDPanel` je implementáciou pracovnej plochy interaktívneho diagramu. Na tejto ploche sa zobrazujú komponenty a manipuluje sa s nimi. Preto je tu implementovaná metóda `paintComponent`, ktorá vykreslí všetky komponenty a akcie používateľa sa sledujú pomocou objektov triedy `Gui.ACDMoudeListener` a `Gui.ACDMouseMotionListener`.

Značnú časť triedy `Gui` tvorí implementácia vytvárania hrán medzi uzlami. Keď je rozhranie v stave ťahania uzlov a ťahané uzly sa pustia nad niektorým iným uzlom, používateľ sa snaží vytvoriť hrany medzi ťahanými uzlami a uzlom, na ktorý uzly hodil. V takomto prípade sa vyvolá metóda `createEdgesToNode`, ktorá zistí, akého typu sú možné hrany medzi uzlami, v prípade nejednoznačnosti sa opýta na správny typ hrany používateľa prostredníctvom objektu typu `EdgeCreatorDialog`.

Pri kliknutí pravým tlačítkom na komponent sa vyvolá objekt triedy `Gui.ComponentPopupMenu`, ktorý umožní používateľovi meniť hodnoty vlastností vybraného komponentu, spúšťať nad komponentom metódy, prípadne komponent zmazať, či skryť.

Trieda `MenuBar` sa týka konceptu globálnych funkcií programu. Globálne funkcie v programe zabezpečuje jednak jadro systému – to sú operácie ukončenia programu, zmazania všetkých komponentov a podobne. Okrem toho globálne funkcie môžu registrovať aj moduly programu.

Každá globálna funkcia musí implementovať rozhranie `javax.swing.Action`. Pomocou tohto rozhrania musí mať nastavené hodnoty s kľúčmi:

- „`_menu`" – text, ktorý určí, v ktorom menu má byť položka s touto akciou umiestnená. Napr. „File", „Edit" a pod.
- „`_priority`" – číslo, ktoré slúži na usporiadanie položiek v menu. Akcie v menu budú zoradené podľa stúpajúcej priority.
- „`_mnemonic`" – číslo (podľa statických hodnôt v triede java.awt.event.KeyEvent), ktoré určuje klávesovú skratku pre túto položku menu.

7.2.6 Zásuvné moduly programu

Táto časť dokumentu nahliada na moduly z pohľadu jadra aplikácie. Uvádza, akým spôsobom jadro s modulmi komunikuje a aké informácie si s ním vymieňa. Pre autorov zásuvných modulov je určená časť 10.

Zásuvný modul programu musí implementovať rozhranie `Module`. Pomocou metód tohto rozhrania si aplikácia od modulu zistí informácie o module (názov, autor, ...) a vypýta si aj registrované typy komponentov a globálne funkcie pomocou metód `getNodeTypes`, `getEdgeTypes`, `getAreaTypes`, `getActions`.

Trieda `ModuleAdapter` je jednoduchá implementácia rozhrania `Adapter`, ktorá definuje premenné, ktoré poskytujú metódy z rozhrania `Adapter`. Teda napríklad definuje premennú name, ktorú vráti implementácia metódy getName. Naplnenie týchto premenných zmysluplnými hodnotami je úlohou programátora odvodenej triedy od `ModuleAdapter`-a – autora zásuvného modulu programu.

Triedy zásuvného modulu (prípadne aj spolu s inými súbormi, ktoré potrebuje, napr. súbormi s ikonami uzlov) môžu byť zbalené do jedného súboru – jar archívu. Na prácu s dátami v jar archíve slúži trieda `JarClassLoader`. Metódy tejto triedy priamo používa objekt `ACD` pri nahrávaní modulu. Okrem toho, aj samotný modul potrebuje vedieť pristupovať k súborom v archíve. K tomu mu slúži metóda `url` triedy `ModuleAdapter`, ktorá vytvorí z relatívnej cesty (názvu súboru) lokátor, pomocou ktorého vedia vstupno/výstupné funkcie javy pracovať so súbormi v jar archíve.

7.3 Implementované moduly

Zásuvné moduly programu ACDesigner, ktoré boli vytvorené v rámci tejto práce boli všetky implementované ako triedy odvodené od triedy `ModuleAdapter`. V konštruktore modulu sa vytvárajú typy komponentov, k nim sú priradené príslušné vlastnosti, metódy a funkcie obsluhy udalostí. Okrem toho sa vytvárajú aj globálne akcie ako triedy odvodené od triedy `AbstractAction`. Tieto vytvorené dáta sa ukladajú do premenných triedy `ModuleAdapter`, ktorá sa stará o samotnú komunikáciu s jadrom programu prostredníctvom rozhrania `Module`.

7.3.1 Modul XML

Zásuvný modul xml implementuje funkcionalitu ukladania a načítavania dát zo súboru a aj funkcionalitu schránky s operáciami kopírovania a vkladania. Formát XML je špecifikovaný v dokumente (BPS00).

Základom tohto modulu je trieda ACDXml, ktorej objekt sa dá skonštruovať buď z dát v aplikácii, alebo zo zadaného súboru. Po skonštruovaní objektu ACDXml sa dáta z neho dajú vložiť do aplikácie, alebo uložiť do súboru. Akcia nahratia dát zo súboru potom vyzerá tak, ako je to naznačené v príklade 1.

Príklad 1 Nahranie dát zo súboru do aplikácie

```
AbstractAction add=new AbstractAction("Merge file") {
  public void actionPerformed(ActionEvent e) {
    fileChooser.showOpenDialog();
    File f=fileChooser.getSelectedFile();
    ACDXml xml=new ACDXml(f);
    xml.load(app);
  }
};
```

Uloženie do súboru je obdobné – objekt `ACDXml xml` sa skonštruuje s argumentom `app` a vyvolá sa metóda `xml.save(f)`.

Funkcionalita schránky je implementovaná tak, že pri kopírovaní do schránky sa do dočasného xml súboru uložia len označené komponenty. Vkladanie skopírovaných dát je jednoduchým načítaním xml dát z dočasného súboru a vloženie dát do aplikácie.

Samotná manipulácia s xml dátami je implementovaná pomocou prístupu *document object model* za použitia tried z balíkov org.w3c.dom a javax.xml.transform.dom (sú štandardnou súčasťou prostredia pre beh aplikácií java).

Do xml súboru sa ukladajú jednak dáta o typoch komponentov a jednak samotné dáta komponentov a ich vlastností. V každom súbore je teda jednoznačne určené komponenty akých typov sú tam uložené a z akých modulov

pochádzajú. Po identifikácii typov komponentov nasledujú samotné dáta komponentov – súradnice uzlov, vrcholy hrán a podobne. Okrem týchto dát sa do súboru ukladajú aj globálne informácie typu veľkosť okna aplikácie a podobne.

Na serializáciu týchto dát sa používajú hodnoty celočíselnej premennej id, ktorá je priradená každému komponentu, vlastnosti a typu komponentu. Pri nahrávaní dát do aplikácie sa však tieto hodnoty nezachovajú zhodné s hodnotami v súbore – mohlo by dôjsť ku kolízii s už existujúcimi komponentami a ich identifikátormi. Na oddelenie identifikátorov sa používa mechanizmus mapovania – vytvoria sa asociatívne tabuľky medzi identifikátormi v súbore a v aplikácii a pomocou nej sa prekladajú identifikátory. Tieto tabuľky sa vytvoria zvlášť pre uzly (`nodeIdMapping`), typy komponentov (`ctIdMapping`) a typy vlastností (`vtIdMapping`).

Presná definícia syntaxe xml súboru je uvedená v súbore *acdesigner-xml.dtd,* ktorý sa nachádza v adresári so zdrojovými kódmi modulu.

7.3.2 Modul MySQL

Zásuvný modul pre tvorbu bezpečnostnej politiky pre MySQL server používa na komunikáciu so serverom rozhranie JDBC implementované konektorom *com.mysql.jdbc.Driver*. JDBC je javovská implementácia špecifikácie na jednotný prístup k databázam ODBC a použitý konektor je dodávaný priamo výrobcom MySQL servera a je voľne dostupný pod GPL licenciou.

V prípade, že sa pri spustení programu nepodarí inicializovať spomínaný JDBC ovládač, nebudú prístupné funkcie importu a exportu bezpečnostnej politiky do MySQL servera, no ináč bude modul funkčný.

Funkcie importu a exportu dát sú implementované vo vnorených triedach `MySqlDataImporter` resp. `MySqlDataExporter`.

Import dát z databázy prebieha postupne – najprv sa načítajú používatelia, zapísaní v tabuľke *user,* potom sa načítavajú dáta z tabuliek *db, table_priv, column_priv*. Pre každý riadok z každej z týchto tabuliek sa najprv overí, či sa používateľ, ktorého sa tento riadok týka nachádza v dátach v ACDesigneri a ak nie, tak sa tento používateľ vytvorí (metóda `loadUserFromRs`). Potom sa k tomuto používateľovi priradí príslušné oprávnenie.

Export dát do databázy prebieha tak, že sa exportujú privilégiá postupne pre každého používateľa (metóda `exportUserPrivs`). V tejto metóde sa najprv overí, či sa už tento používateľ nachádza v tabuľke *user* a potom sa prechádzajú všetky jeho privilégiá a zapisujú sa do príslušnej tabuľky.

7.3.3 Modul Unix

V zásuvnom module pre návrh politiky založenej na bezpečnostnom modeli systému Unix sa konfigurácia importuje z textového súboru, ktorý je možné vytvoriť

príkazom `ls -lR`.

Zoznam používateľov a skupín sa importuje a exportuje do súborov passwd a group s príslušnou syntaxou.

Pri vzťahoch „login group" a „group member" sa používajú metódy na obsluhu udalostí typu „add" na rozlíšenie, či sa má pri vytvorení vzťahu medzi používateľom a skupinou vytvoriť vzťah jedného alebo druhého typu. V prípade, že používateľ ešte nemá priradenú žiadnu skupinu ako „login group", vráti obsluha udalosti „add" pre vzťah „login group" hodnotu „true" a pre vzťah „group member" hodnotu „false". V prípade, že už daný používateľ má zadanú „login group", vrátia obsluhy udalostí opačné hodnoty. Takýmto spôsobom sa automaticky rozhodne, ktorý z dvoch možných typov vzťahov sa medzi uzlami vytvorí.

Okrem toho, pri vytvorení nového používateľa sa mu automaticky nastaví chybový stav (`setError(true)`), pretože tento vytvorený používateľ ešte nemá nastavenú žiadnu skupinu ako „login group". Tento chybový stav sa potom aktualizuje v metódach na obsluhu udalostí vzťahu „login group".

Importné vnútorné triedy modulu `Unix` používajú na analyzovanie obsahu súborov regulárne výrazy implementované v balíku `java.util.regex`.

Na import súborovej štruktúry – adresárov, súborov a ich priradenia používateľom a skupinám spolu s určením ich prístupových práv sa používa vnútorná trieda `FilesImporter` a jej metóda `importFiles`. Táto metóda slúži na import prístupových práv k jednému adresáru do hĺbky jednej úrovne. Táto metóda dostane ako svoj argument adresár `parent`. V prípade, že `parent` má hodnotu `null`, importuje sa koreňový adresár. Na vyhľadávanie príslušných záznamov pre daný adresár a potom aj na zisťovanie údajov o konkrétnom súbore resp. podadresári sa používajú regulárne výrazy.

Pri importovaní konkrétneho adresára sa dodržiava rozmiestnenie uzlov na ploche diagramu. Po „rozbalení" adresára sa všetky uzly, ktoré boli zobrazené pod týmto uzlom posunú nadol tak, aby sa vytvárala stromová štruktúra súborového systému. Na to slúži metóda `moveTreeDown`.

7.3.4 Modul RBAC

Zásuvný modul pre návrh politiky založenej na RBAC bezpečnostnom modeli nekomunikuje so žiadnym konkrétnym systémom na implementáciu vytvorenej politiky. Preto je jeho zdrojový kód krátky a čitateľný. Modul nedefinuje žiadne globálne funkcie programu a definuje len typy komponentov, ich vlastnosti a metódy.

Pri metódach na označenie všetkých uzlov vo vzťahu s nejakým uzlom sa používajú metódy Node.nodesConnectedToIterator a nodesConnectedFromIterator. Tieto vrátia iterátor, ktorým sa môžu prechádzať všetky uzly, ktoré sú

v danom vzťahu s označeným uzlom a potom sa tieto uzly vyznačia. Načrtnutie takéhoto postupu pre metódu vyhľadania všetkých rolí používateľa je v príklade 2.

Príklad 2 Metóda vyhľadanie všetých rolí používateľa

```
u.addMethod(new Method("all roles") {
  body(Component c) {
    Node node = (Node)c;
    Iterator i=node.nodesConnectedToIterator(ua);
    while(i.hasNext()) {
      Node toNode=(Node) i.next();
      toNode.select();
    }
  }
});
```

8 Overenie riešenia

Táto časť práce hovorí o verifikácii a validácii vytvoreného systému. Zhodnotením systému sa zaoberá najmä časť 11.

V špecifikácii požiadaviek (v časti 4) boli požiadavky na vytváraný systém rozdelené do niekoľkých tried. Podľa tých istých tried je opísaná aj verifikácia ich splnenia, uvedená v nasledujúcich podkapitolách.

8.1 Zobrazenie komponentov

Požadované bolo, aby sa dalo identifikovať:

- Typ komponentu – typ uzla je označený obrázkom (ikonou). Typ vzťahu môže byť vyznačený farbou hrany alebo oblasti.
- Identita komponentu – identita uzla je označená textom priamo pod ikonou uzla. Identitu hrany a oblasti je možné identifikovať v kontextovom menu pre tieto komponenty.
- Iné vlastnosti komponentu – je možné ich zadať v kontextovom menu komponentu.
- Dočasné atribúty – objekt označený používateľom sa označí orámovaním (uzol) alebo označením krajných bodov (hrana, oblasť). Objekty označené systémom sú označené rovnako – systém nerozoznáva medzi spôsobmi označenia komponentov. Presúvané objekty sú vyznačené sivou farbou pozadia.

8.2 Všeobecné požiadavky na zobrazovanie

Jedná sa o požiadavky na používateľské rozhranie systému:

- Prirodzenosť a intuitívnosť – prostredie programu simuluje papier a ceruzku (napríklad aj takými detailami ako sú biele pozadie a tmavé komponenty, tvar ikon má pripomínať rukou kreslené tvary a podobne). Všetky vlastnosti komponentu a možné operácie s komponentom (metódy) sú prístupné priamo pri komponente v kontextovom menu. Vytváranie vzťahov medzi dvoma uzlami je prirodzene umožnené ťahaním komponentu na iný komponent, kedy sa automaticky naznačí vytváraná hrana. Označovanie oblastí je možné voľným kreslením pomocou ťahania myšou.
- Konzistentnosť – jedným z príkladov konzistentnosti je rovnaký tvar ikony pre používateľa vo všetkých implementovaných moduloch.

- Ergonómia – kontrastnosť zobrazenia je odstupňovaná v niekoľkých stupňoch:

  1. nezobrazenie – pre skryté komponenty,
  2. sivé zobrazenie – pre prenášané uzly (tie sú výraznejšie tým, že reagujú na pohyby myši, takže ich samotné zobrazenie môže mať menší farebný kontrast),
  3. normálne zobrazenie,
  4. chybové zobrazenie – komponent je označený výraznou červenou farbou.

- Použiteľnosť podľa uznávaných odporúčaní pre tvorbu používateľského rozhrania – použiteľnosť má nedostatky najmä preto, že absentuje možnosť opravy chyby používateľa – neexistencia funkcie typu „Undo".

- Jednoduchosť a prehľadnosť – systém sa snaží o sprehľadnenie zobrazeného diagramu najmä pri importoch dát – snaží sa objekty umiestňovať na plochu tak, aby sa navzájom neprekrývali, aby sa hrany čo najmenej krížili a podobne. Prehľadnosť podľa subjektívnych názorov autora systému a niekoľkých používateľov nie je úplne uspokojujúca.

8.3 Akcelerácia návrhu politiky

Vytvorený systém mal za úlohu akcelerovať tieto etapy návrhu bezpečnostnej politiky:

- Identifikácia objektov a subjektov politiky – moduly pre jednotlivé prostredia umožňujú import údajov o objektoch a subjektoch politiky. Tieto komponenty je následne možné voľne myšou presúvať (jednotlivo alebo v skupinkách).

- Definovanie hierarchie rolí – systém umožňuje definovať vzťahy nadradenosti a podradenosti, je možná aj implementácia kontroly na obmedzenia (pomocou mechanizmu metód obsluhy udalostí).

- Priradenie používateľov a oprávnení – v tejto etape boli požiadavky najmä na prehľadnosť – identita každého objektu je označená priamo pod ikonou, je možné presúvanie a preusporiadavanie objektov aj spolu s ich vzťahmi. Na zorientovanie sa používateľa je možné použiť metódy na označenie napr. všetkých rolí používateľa (pomocou mechanizmu metód komponentov).

- Kontrola obmedzení je možná automaticky pomocou obsluhy udalostí alebo na vyžiadanie pomocou metód komponentov. Aplikácia môže buď zabrániť vykonaniu akcie, ktorá by viedla k porušeniu obmedzení (keď metóda obsluhy udalostí vráti neúspech) alebo je možné označiť chybový stav komponentu.
- Verifikácia politiky – je možné implementovať (a v existujúcich moduloch niektoré aj boli implementované) metódy, ktoré označia napríklad všetky oprávnenia používateľa a podobne.

8.4 Operácie s komponentami

Komponenty môže používateľ pridávať z palety, presúvať ich, či mazať. Môže editovať ich parametre v kontextovom menu. Môže dať skryť vybrané komponenty (jednotlivo alebo v skupinách alebo aj všetky komponenty daného typu). Vytváranie a mazanie vzťahov medzi komponentmi sa deje vytváraním a mazaním hrán a oblastí diagramu. Nad označenou skupinou komponentov sa dajú vykonať akcie hromadného zmazania alebo skrytia komponentov. Vyhľadávanie komponentov nebolo implementované z nedostatku času. Automatické rozmiestnenie komponentov na ploche sa deje pri každom importovaní dát z prostredia.

8.5 Použiteľnosť v reálnych aplikáciách

Vytvorené moduly umožňujú navrhovať politiku

- založenú na modeli RBAC. Jedná sa o všeobecný modul, bez komunikácie s nejakými konkrétnymi bezpečnostnými mechanizmami;
- pre systém Unix. Tento modul nepriamo komunikuje s bezpečnostnými mechanizmami operačného systému;
- pre databázový server MySQL s priamou obojsmernou väzbou na systém privilégií konkrétneho servera.

8.6 Modulárnosť riešenia

Mechanizmom zásuvných modulov je možné, aby programátori prispôsobovali systém ACDesigner na návrh bezpečnostných politík iných systémov. Rozhranie pre zásuvné moduly je presne definované a dokumentované jednak tutoriálovým spôsobom (kapitola 10) a aj referenčnou príručkou (javadoc-dokumentácia na priloženom dátovom médiu).

8.7 Programová manipulácia s komponentmi

Zásuvný modul má možnosť definovať typy komponentov, vlastností, metód a funkcií obsluhy udalosti pridania a zmazania komponentu. V takýchto metódach má modul priamy prístup aj k jednotlivým komponentom, s ich dátami môže bez obmedzenia manipulovať.

9 Používateľská príručka

9.1 Inštalácia a spustenie

Systém ACDesigner sa skladá z jadra programu a zo zásuvných modulov. Jadro programu je v súbore ACDesigner.jar a každý modul je vo zvláštnom súbore s príponou .jar.

Na spustenie systému ACDesigner je potrebné mať nainštalované prostredie pre beh aplikácií Java Runtime Engine verzie 1.4 alebo vyššej. Jednotlivé moduly môžu vyžadovať ešte prípadné ďalšie komponenty, jadro ACDesignera nepoužíva žiadne.

Pri spustení programu je potrebné zaviesť aspoň jeden zásuvný modul. Spustenie programu spolu so zásuvným modulmi sa vykoná príkazom:

```
java -jar ACDesigner.jar <modul.jar> ...
```

Napríklad spustenie programu a zavedenie modulov pre xml a MySQL sa vykoná príkazom:

```
java -jar ACDesigner.jar XMLModule.jar MysqlModule.jar
```

Modul xml sa odporúča použiť vždy, pretože umožňuje používať funkcie pre prácu so súbormi – ukladanie, nahrávanie dát zo súboru a poskytuje aj možnosti práce so schránkou – kopírovanie a vkladanie. V ďalších častiach používateľskej príručky sa vždy predpokladá, že je modul xml použitý ak nie je uvedené inak.

9.2 Základy práce s programom

Okno grafického rozhrania systému ACDesigner je vertikálne rozdelené na časti

- lišta s menu,
- lišta s nástrojmi a komponentmi,
- pracovná plocha diagramu.

Centrom práce so systémom ACDesigner je pracovná plocha interaktívneho diagramu – biela plocha zaberajúca najväčšiu časť okna aplikácie. Na tejto pracovnej ploche sa tvorí vizuálny model navrhovanej bezpečnostnej politiky vo forme diagramu. V diagrame vystupujú:

- uzly, zobrazené ako ikony,
- vzťahy medzi dvomi uzlami, zobrazené ako čiara medzi uzlami,
- vzťahy medzi skupinou uzlov, zobrazené ako farebné oblasti.

Význam ikon a vzťahov medzi nimi je závislý od použitého zásuvného modulu – pre každý modul budú ikony a vzťahy iné. Použitie jednotlivých modulov je vysvetlené v ďalších častiach používateľskej príručky.

Na lište s komponentmi sa nachádzajú tlačidlá. Prvé tlačidlo zľava je vždy tlačidlo so šípkou. Toto tlačidlo slúži ako manipulačný nástroj. Keď ho stlačíte, môžete potom myšou manipulovať s komponentmi na pracovnej ploche:

- presúvať ich ťahaním,
- označovať jednotlivo stlačením ľavého tlačítka na ikone, čiare alebo oblasti,
- označovať skupinu stlačením *Control* a ľavého tlačítka,
- označovať skupinu ťahaním myšou so stlačeným ľavým tlačítkom (označia sa ikony zahrnuté do obdĺžnikového výberu),
- vyvolávať kontextové menu stlačením pravého tlačítka myši na ikone, čiare alebo oblasti. Kontextové menu obsahuje niekoľko sekcií:
  - sekcia vlastností – kliknutím na niektorú vlastnosť môžete zmeniť napríklad meno používateľa, meno adresára a podobne podľa toho, aké vlastnosti má príslušný uzol, hrana alebo oblasť,
  - sekcia metód – kliknutím na metódu sa vyvolá akcia nad zvoleným uzlom, čiarou alebo oblasťou. Môžete tak napríklad dať vyznačiť všetky súbory v adresári, dať vyznačiť všetky ikony v oblasti a podobne podľa toho, aké metódy zvolený komponent poskytuje
  - ostatné – tu sa nachádzajú akcie na trvalé zmazanie alebo na dočasné skrytie zvolenej ikony, čiary alebo oblasti

Ako ďalšie tlačidlá na lište sú tlačidlá s ikonami. Keď stlačíte tlačidlo s ikonou, po každom kliknutí na plochu diagramu pridáte novú ikonu na miesto, kde ste klikli.

Ďalšie tlačidlá sú tlačidlá s textovým popisom[3]. Tieto slúžia na vytváranie oblastí. Kliknite na tlačidlo oblasti a potom stlačením a ťahaním myši nad oblasťou pracovnej plochy vytvoríte súvislú farebnú oblasť. Oblasť reprezentuje vzťah medzi ikonami, ktoré sa v nej nachádzajú. Oblasť je možné zväčšovať, zmenšovať, alebo upravovať jej hranice tak, že pri zapnutom manipulačnom tlačidle (tlačidlo so šípkou) kliknete na oblasť, čím ju označíte a jej vrcholové body sa označia malými čiernymi štvorčekmi. Tento štvorček môžete myšou ťahať a tým meniť hranicu oblasti.

[3] V závislosti od použitého zásuvného modulu tam môžu alebo nemusia byť.

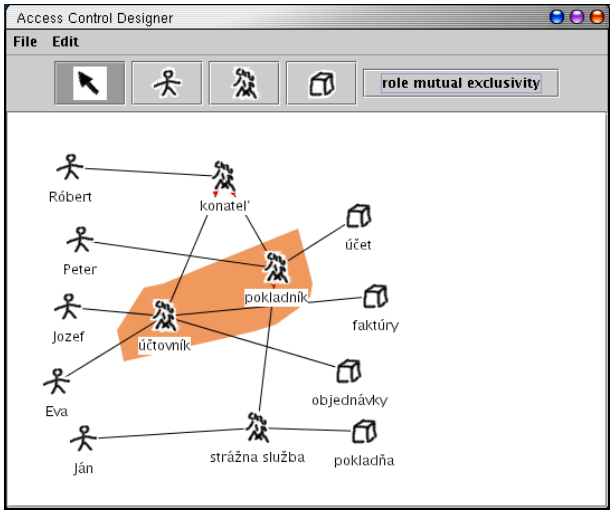
Čiary medzi ikonami, ktoré znázorňujú vzťah medzi dvoma ikonami vytvoríte tak, že pri zapnutom manipulačnom tlačidle myšou chytíte a ťaháte ikonu, alebo označenú skupinu ikon a pustíte ju na niektorú inú ikonu. V tomto momente sa buď automaticky vytvoria čiary medzi označenou ikonou (ikonami) a ikonou, na ktorú ste ikony hodili. V prípade, že je možných viac typov čiar medzi dvojicou ikon, systém sa vás dialógom opýta, ktorú hranu chcete vytvoriť.

Keď kliknete pravým tlačítkom myši na tlačidle s ikonou alebo oblasťou, môžete skryť alebo zobraziť všetky ikony resp. oblasti daného typu.

Vrchná lišta obsahuje rôzne menu. V menu „File" nájdete príkazy na uloženie do súboru, natiahnutie zo súboru, zmazanie aktuálne vytváranej politiky a podobne. V menu „Edit" sa nachádzajú príkazy na prácu so schránkou – kopírovanie a vloženie. Schránka funguje aj medzi dvomi inštanciami ACDesignera – ak do nej skopírujete nejaké komponenty v jednej inštancii, v druhej ich môžete vložiť.

9.3 Návrh bezpečnostnej politiky založenej na RBAC

Zásuvný modul pre návrh politiky založenej na modeli RBAC je všeobecný a neviaže sa na žiadny konkrétny systém či bezpečnostné mechanizmy. Na obrázku 13 je typická obrazovka ACDesignera pri práci s modulom RBAC.



Obrázok 13: Modul RBAC

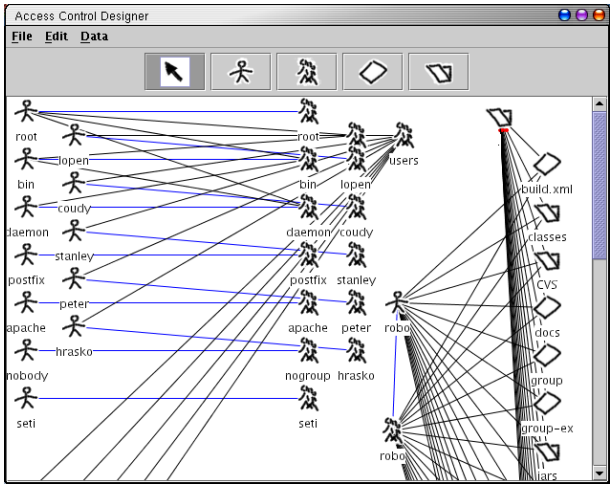
Ako ikony (uzly diagramu) tu vystupujú používatelia, role a oprávnenia. Každý z týchto komponentov má priradené meno. Vzťahy priradenia používateľov do role, priradenia oprávnenia role a hierarchie rolí sú reprezentované hranami medzi príslušnými uzlami. Vzťah vzájomného vylučovania medzi rolami je reprezentovaný oblasťou.

9.4 Návrh bezpečnostnej politiky pre OS Unix

Zásuvný modul pre návrh politiky pre Unix interpretuje reálne dáta z operačného systému. Ako uzly tu vystupujú:

- používatelia, ktorí sa importujú zo súboru /etc/passwd,
- skupiny, ktoré sa importujú zo súboru /etc/group,
- súbory a adresáre, ktoré sa importujú zo súboru, ktorý obsahuje výstup unixového príkazu ls -lR, vykonaného v adresári, pre ktorý sa má navrhovať prístupová politika.

Ukážka obrazovky programu ACDesigner pri práci s modulom pre Unix je na obrázku 14.



Obrázok 14: Modul Unix

Vzťahy medzi používateľmi a skupinami sú dva rôzne – jeden vzťah (reprezentovaný modrou čiarou) je vzťah „login group". Je to skupina, ktorú má používateľ zapísanú v súbore /etc/passwd. Pre každého používateľa musí existovať práve jedna, takže ak nejaký používateľ nemá priradenú žiadnu takúto skupinu, je označený výraznou červenou farbou. Viac ako jedna takáto skupina sa jednému používateľovi priradiť nedá – ak sa má vytvoriť ďalšia hrana medzi používateľom a nejakou skupinou, vytvorí sa hrana typu „group member". To je skupina, ku ktorej je používateľ zapísaný v súbore /etc/group. Tento vzťah je reprezentovaný hranou čiernej farby a jeden používateľ ich môže mať neobmedzený počet. Načítanie aktuálnej konfigurácie zo súborov sa dá vykonať príkazom „Import users and groups" v menu „Data".

Štruktúra súborov a adresárov je reprezentovaná uzlami súborov a adresárov, spolu so vzťahmi „súbor je v adresári" a „je podadresárom". V menu „Data"

sa nachádza príkaz „Import file structure", ktorá zo zadaného súboru načíta do stromovej štruktúry nielen adresáre a súbory, ale aj ich priradenie existujúcim používateľom a skupinám (odporúčaný postup je najprv importovať používateľov a skupiny a potom importovať štruktúru súborov). Po skončení práce smodelom je možné vyexportovať vytvorenú politiku vo forme skriptu pomocou položky „Export file structure" v menu Data.

Prístupové práva vlastníka súboru, skupiny a ostatných sú reprezentované takto:

- práva vlastníka sú zahrnuté ako vlastnosti hrany medzi používateľom a súborom (alebo adresárom). Stačí kliknúť pravým tlačítkom na príslušnú čiaru a v hornej časti kontextového menu vypraf príslušný prístupový typ (čítanie, zápis, vykonanie),
- práva skupiny sú vlastnosťami hrany medzi skupinou a súborom. Obdobne stačí kliknúť pravým tlačítkom myši na čiaru a v menu vybrať prístupový mód,
- práva ostatných sú zahrnuté ako vlastnosti samotného súboru. Keď vyvoláte kontextové menu súboru alebo adresára, medzi vlastnosťami nájdete vlastnosti ako „others read", „others write" a „others execute".

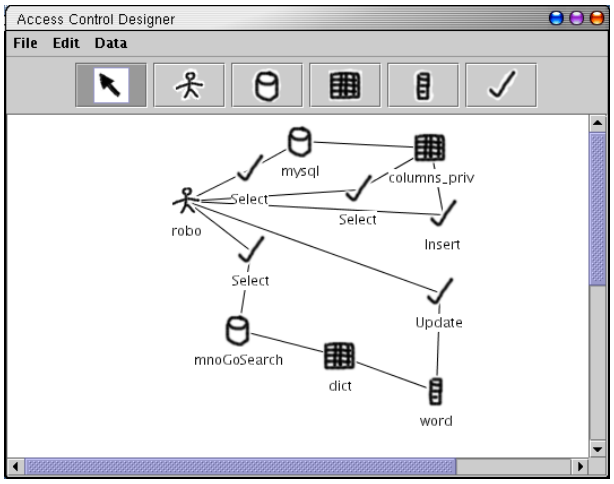
9.5 Návrh bezpečnostnej politiky pre MySQL server

Zásuvný modul pre návrh politiky pre databázový server MySQL komunikuje priamo s týmto serverom. Z tohto dôvodu tento modul má aj špeciálne požiadavky – pre funkčnosť importu a exportu je potrebné, aby bol do prostredia pre beh java-aplikácií nainštalovaný ovládač pre MySQL databázu. Konkrétne je vyžadovaný ovládač `com.mysql.jdbc.Driver`, ktorý sa nachádza aj na médiu priloženom k tejto práci a je možné ho zdarma stiahnuť z webového sídla autorov databázy (MYSQL).

Na obrázku 15 je typická obrazovka pri práci s modulom MySQL.

Ako ikony (uzly diagramu) tu vystupujú:

- používateľ – jeho vlastnosti sú prihlasovacie meno, adresa počítača, z ktorého sa k serveru prihlasuje,
- databáza,
- tabuľka, priradená hranou ku databáze,
- stĺpec tabuľky, priradený hranou ku svojej tabuľke,
- oprávnenie.



Obrázok 15: Modul MySQL

Oprávnenie môže byť spojené s databázou, alebo tabuľkou či stĺpcom. V takomto prípade sa toto oprávnenie vzťahuje len na ten objekt, s ktorým je spojené. V prípade, že oprávnenie nie je spojené so žiadnym takýmto objektom, jedná sa o serverové (superužívateľské) oprávnenie, vzťahujúce sa na všetky objekty. Bližšie o systéme privilégií databázy MySQL hovorí časť 3.3.

V menu „Data" sa nachádzajú položky „Import data from mysql server" a „Export data to mysql server". Tieto slúžia na komunikáciu s databázovým serverom. V prípade, že sú tieto položky v menu neaktívne, systému ACDesigner sa nepodarilo inicializovať ovládač pre MySQL. V takom prípade sa presvedčte, či je správne nainštalovaný.

Po spustení importu sa vás systém opýta na údaje o databázovom serveri, z ktorého chcete importovať (adresu servera, prihlasovacie meno a heslo). Keď sa mu podarí pripojiť, naimportuje sa celá štruktúra prístupových práv servera – používatelia, databázy, tabuľky, stĺpce a priradenie oprávnení.

Keď vykonáte požadované úpravy prístupových práv, môžete použiť exportnú funkciu, ktorá sa takisto opýta na server, na ktorý chcete vytvorenú politiku aplikovať a keď sa mu podarí pripojiť, aplikuje ju.

10 Programátorská príručka pre tvorcov zásuvných modulov

Táto časť dokumentu má za cieľ pomôcť programátorom aplikovať ACDesigner na návrh bezpečnostných politík pre iné systémy pomocou zásuvných modulov.

Príručka však hovorí len o spôsobe tvorby modulu ako takého. Na to, aby modul mohol manipulovať s dátami v aplikácii je potrebné, aby programátor zásuvného modulu poznal spôsob reprezentácie a ukladania dát v programe. O tom hovorí všeobecnejšie časť o návrhu systému (6) a časť o implementácii (7). Presné informácie o implementácii sú uvedené v dokumentácii k zdrojovým kódom (javadoc-dokumentácii), ktorá je umiestnená aj na priloženom elektronickom médiu. Okrem toho je možné nahliadnuť aj do zdrojových kódov implementovaných modulov a aj do kódu jadra aplikácie (tieto zdrojové kódy je tiež možné nájsť aj na priloženom médiu).

10.1 Princíp zásuvných modulov

Zásuvný modul systému ACDesigner je samostatne kompilovaný kód, ktorý si ACDesigner počas svojho behu načíta a vykonáva kód v ňom umiestnený.

Zásuvný modul má za úlohu vytvoriť a do programu zaregistrovať typy komponentov spolu s príslušnými typmi hodnôt, metódami a funkciami obsluhy udalostí. Okrem toho môže obsahovať aj implementáciu globálnych funkcií programu ACDesigner, ktoré sú používateľovi prístupné z menu.

Kód modulu je umiestnený v jednej alebo viacerých triedach. Jedna z tried modulu musí implementovať rozhranie pre moduly, ktoré definuje ACDesigner. Táto trieda je označená ako hlavná trieda modulu.

Okrem toho môžu byť súčasťou modulu aj iné súbory ako len triedy, napríklad súbory s obrázkami ikon, konfiguračné súbory a podobne. Všetky súbory modulu je možné zbaliť do jedného archívu, ktorý sa jednoducho distribuuje a používa.

10.2 Rozhranie pre moduly

Rozhranie `Module` definované v jadre systému ACDesigner obsahuje metódy, pomocou ktorých jadro komunikuje s modulom. V časti 7.2.6 sme hovorili o pohľade na toto rozhranie z pohľadu jadra.

Po vytvorení objektu, ktorý implementuje rozhranie `Module` jadro zavolá metódu `setApplication`, ktorou odovzdá modulu ako argument referenciu na bežiacu aplikáciu (objekt triedy `ACD`). Pomocou tejto referencie potom môže modul manipulovať s dátami aplikácie.

Metódy rozhrania `getNodeTypes`, `getEdgeTypes` a `getAreaTypes`, ktoré modul implementuje musia vracať kolekcie (objekty triedy java.util.Collection), v ktorých sú

uložené objekty príslušných typov komponentov (objekty typu `NodeType`, `EdgeType` resp. `AreaType`), ktoré chce modul do systému zaviesť.

Metóda rozhrania `getActions` má vracať kolekciu objektov typu javax.swing.Action, ktoré reprezentujú globálne funkcie, ktoré chce modul do systému zaviesť. O spôsobe použitia triedy `Action` v ACDesigneri hovorí bližšie časť 7.2.5.

Okrem toho rozhranie `Module` definuje aj metódy `getName`, `getDescription` a `getAuthor`, ktoré majú vracať názov modulu, jeho opis a informácie o autorovi modulu.

10.3 Adaptér rozhrania

Trieda `ModuleAdapter` je jednoduchá implementácia rozhrania `Module`, ktorá má za úlohu uľahčiť prácu programátorom zásuvných modulov programu ACDesigner. Táto trieda obsahuje premenné, ktoré slúžia na ukladanie vytvorených typov komponentov a tried obsluhujúcich globálne funkcie programu.

Činnosť programátora modulu sa potom môže obmedziť na vytvorenie odvodenej triedy od triedy `ModuleAdapter` a naplnenie týchto premenných počas behu konštruktora tejto triedy. Implementácia metód rozhrania `Module` v triede `ModuleAdapter` potom poskytne tieto naplnené dáta jadru programu bez toho, aby tieto metódy musel programovať autor modulu, pretože odvodená trieda ich implementáciu zdedí od `ModuleAdapter`a.

Premenné na ukladanie typov komponentov sú `nodeTypes`, `edgeTypes` a `areaTypes`, všetky sú typu `ComponentTypes`, teda kontajnery na typy komponentov. Globálne funkcie reprezentované objektami triedy `Action` sa ukladajú do kolekcie `actions`. Meno modulu, jeho opis a informácie o autorovi sa ukladajú do premenných `name`, `description` a `author`.

Implementácie metódy `setApplication` uloží referenciu na aplikáciu do premennej `app`, ktorú potom môže modul používať na prístup k dátam v aplikácii.

Trieda `ModuleAdapter` obsahuje aj funkciu `url`, ktorá slúži na sprostredkovanie prístupu modulu k súborom patriacim k modulu, napríklad súborom s obrázkami ikon a podobne. Metóda `url` z relatívnej cesty k súboru vytvorí správnu absolútnu cestu k danému súboru aj s ohľadom na to, že tento súbor môže byť prípadne uložený v jar-archíve spolu s binárnymi súbormi tried modulu.

10.4 Odporúčaný spôsob tvorby zásuvného modulu

V príkladoch 3 a 4 sú uvedené ukážky kódu implementácie modulov.

Programátor môže pri implementácii zásuvného modulu programu Access Control Designer postupovať týmto spôsobom:

1. Vytvoriť hlavnú triedu modulu ako podtriedu triedy `ModuleAdapter`.
2. Naprogramovať bezargumentový konštruktor triedy, ktorý vytvorí požadované typy komponentov, priradí im typy vlastností, metódy a funkcie obsluhy udalostí.

Príklad 3 Príklad implementácie modulu – vytvorenie dátových typov

```
public class RBACModule
  extends ModuleAdapter
{
  public RBACModule() {
    NodeType u,r,p;
    u=new NodeType("user", new ImageIcon(url("images/user.png")));
    u.addValueType(new ValueType("Username", "string"));
    u.addMethod(new Method("all roles") {
      public boolean body(Component c) {
...
```

3. Vytvorené dátové typy priradiť do príslušných kontajnerov triedy `ModuleAdapter`.
4. Vytvoriť objekty triedy `javax.swing.AbstractAction` a k nim implementovať funkcie obsluhy udalostí (`java.awt.event.ActionEvent`), ktoré budú vykonávať globálne funkcie poskytované modulom.

Príklad 4 Príklad implementácie modulu – vytvorenie globálnej funkcie

```
public class XMLModule
  extends ModuleAdapter
{
  public XMLModule() {
    AbstractAction add=new AbstractAction("Add file") {
      public void actionPerformed(ActionEvent e) {
        JFileChooser fileChooser=new JFileChooser();
...
      }
    }
    add.putValue(„menu’’, „File’’);
  }
}
```

5. Vytvorené globálne funkcie (objekty typu Action) vložiť do kontajnera actions.

Po skompilovaní zdrojového kódu modulu vzniknú súbory typu .class. Tieto je možné používať s ACDesignerom priamo – zadať meno hlavnej triedy modulu ako argument ACDesignera a v takomto prípade bude ACDesigner hľadať

príslušné binárne súbory .class na miestach, ktoré sú uvedené v premennej prostredia CLASSPATH, kde je zvyčajne aj aktuálny adresár.

Na zjednodušenie distribúcie modulu je možné zbaliť všetky binárne súbory tried modulu spolu s ostatnými potrebnými súbormi do jedného súboru – jar archívu. V takomto prípade je potrebné, aby v súbore manifest bola ako `Main-Class` uvedená hlavná trieda modulu.

Napríklad keď je hlavnou triedou modulu trieda RBACModule, binárne súbory modulu sa nachádzajú v adresári classes a obrázky s ikonami v adresári images, bude obsah súboru Manifest takýto:

```
Main-Class:  RBACModule
```

a jar archív modulu RBACModule.jar vytvoríme príkazom:

```
jar cmf Manifest RBACModule.jar classes images
```

Potom pri spustení ACDesignera s týmto modulom stačí zadať ako argument meno tohto jar-archívu.

11 Zhodnotenie projektu

Primárnym cieľom vizuálneho nástroja na návrh bezpečnostnej politiky – Access Control Designer je prekonať problémy pri návrhu bezpečnostných politík. Používateľ pomocou tohto nástroja zachytí požiadavky na politiku v grafickej notácii vychádzajúcej z všeobecne akceptovaného bezpečnostného modelu a tento nástroj mu vygeneruje konfiguráciu bezpečnostných mechanizmov, ktoré chce na jej implementáciu použiť. Túto konfiguráciu je potom možné priamo alebo nepriamo aplikovať na cieľový systém a tak nasadiť vytvorenú bezpečnostnú politiku do prevádzky.

Podľa výsledkov analýzy požiadaviek na funkcionalitu takéhoto nástroja a požiadaviek na pohodlnú, efektívnu a prirodzenú prácu používateľa sme navrhli a implementovali všeobecný nástroj na vizuálny návrh bezpečnostnej politiky. Jeho možnosti sme demonštrovali na modeli bezpečnostnej politiky RBAC, na bezpečnostnom modeli operačného systému Unix a na bezpečnostnom modeli databázového servera MySQL.

Diplomová práca tesne nadväzuje na diplomový projekt (T02), ktorý autor riešil v predchádzajúcich semestroch štúdia. Počas práce na diplomovom projekte vznikol koncept modulárneho nástroja, prvotná analýza a návrh takéhoto nástroja a vznikla aj prvotná verzia jadra systému Access Control Designer. Počas vypracovávania diplomovej práce pokračovala analýza a návrh riešenia doplnená o spätnú väzbu zo skúseností z práce s prototypovou implementáciou. Ďalej pokračovala implementácia modulov systému pre tvorbu bezpečnostných politík pre konkrétne prostredia – všeobecný model založený na RBAC, modul pre politiku OS Unix s možnosťou nepriamej komunikácie s konkrétnym prostredím a bezpečnostými mechanizmami a modul pre server MySQL, ktorý priamo komunikuje a riadi bezpečnostné mechanizmy v bežiacom databázovom serveri.

Cieľom práce bolo, aby vytváraný produkt mal dva hlavné atribúty: univerzálnosť a vizuálny prístup.

Čo sa týka univerzálnosti, bol vytvorený koncept všeobecného interaktívneho diagramu, rozširovateľného pomocou zásuvných modulov. Implementácia tiež spĺňa všetky požiadavky na rozširovateľnosť riešenia a možnosť aplikácie na rôzne systémy.

Keď si raz používateľ zvykne na koncept práce s ACDesignerom, je pre neho jednoduché bez problémov prechádzať medzi návrhom politík pre rôzne systémy. Konkrétny používateľ s priemernými počítačovými znalosťami a bez znalostí bezpečnostných modelov, potom ako pochopil spôsob navrhovania RBAC politiky pomocou ACDesignera, v priebehu niekoľkých minút bol schopný začať pracovať s politikou pre MySQL server. Aj toto je jeden z rozmerov efektivity.

Atribút vizuálnosti vytvoreného riešenia sme sa snažili dosiahnuť tak, že sme umožnili používateľovi systému navrhovať konkrétne, presné vzťahy v bezpečnostnej politike manipuláciou s objektami interaktívneho diagramu. Ukázali sme, že

- takýto prístup je možný a má zmysel o ňom diskutovať,
- výhodou takéhoto prístupu k návrhu bezpečnostnej politiky je najmä to, že pre človeka je prirodzené manipulovať s objektami a vzťahmi medzi nimi,
- nevýhodou oproti systémom dialógového typu s presnou štruktúrou používateľského rozhrania (tabuľky, zoznamy, stromy atď.) je menšia prehľadnosť modelu. Hoci sme sa snažili pri vytváraní systému čo najviac sprehľadňovať zobrazovaný model politiky, stále si myslíme, že prehľadnosť tohto prístupu nie je ideálna.

Ďalšie možné rozširovanie vytvoreného produktu vidíme v dvoch smeroch: vylepšovanie jadra ACDesignera a pridávanie nových zásuvných modulov.

Úpravy jadra by mohli byť smerované na vylepšenie použiteľnosti programu (napríklad doplnením možnosti vrátenia vykonaných akcií používateľa, zvýšením prehľadnosti zobrazovaného modelu a podobne) a aj pridávaním nových funkcií jadra (napríklad zväčšovanie a zmenšovanie, editovanie hodnôt priamo na ploche diagramu a podobne).

Hodnota vytvoreného systému a jeho použiteľnosť v reálnom prostredí závisí od miery akceptovania spôsobu práce s modelom, ktorý implementuje jadro systému ACDesigner a od kvality a funkčnosti vytvorených zásuvných modulov.

Ku akceptovaniu jadra môžeme napríklad uviesť, že pre rok 2003/04 bol na Katedre informatiky a výpočtovej techniky FEI STU vypísaný záverečný projekt, ktorého zadaním je vytvoriť zásuvný modul ACDesignera pre návrh bezpečnostnej politiky pre bezpečnostný systém Medusa DS9.

Využiteľnosť všeobecných modulov (napr. implementovaného modulu pre RBAC politiky) je najmä pri vzdelávaní ako modelovací a predvádzací nástroj, prípadne pre prezentovanie konceptov a návrhov politík na úrovni konceptu.

Využitie modulu pre XML je nielen pre priebežné ukladanie aktuálneho stavu práce s modelom, ale aj možnosť komunikácie medzi viacerými používateľmi vymieňaním si vytvorených xml súborov. Použitie otvoreného formátu dát a definovanie jeho štruktúry pomocou dtd súboru umožňuje aj prípadné jednoduché použitie súborov vytvorených ACDesignerom v iných systémoch.

Účel modulov s reálnym rozhraním na iné systémy (napr. MySQL modulu) je použitie na návrh reálnej bezpečnostnej politiky pre tieto systémy. Implementovaný zásuvný modul pre MySQL to umožňuje.

A Použitá literatúra

(BC98) J. Barkley, A. Cincotta: Managing Role/Permission Relationships Using Object Access Types. Third ACM Workshop on Role Based Access Control, 1998.
`http://hissa.ncsl.nist.gov/rbac/rgperms/rgperms.htm`

(BEN02) C. Benson, A. Elman, S. Nickell et al.: Gnome User Interface Guidelines. GNOME Usability Project, 2002. Online verzia:
`http://developer.gnome.org/projects/gup/hig/`

(BLP76) D.E. Bell, L.J. LaPadula: Secure computer system: Unified exposition and multics interpretation. Technical Report MTR-2997, Mitre Corp., Bedford Massachusetts, USA, 1976.

(BPS00) T. Bray, J. Paoli, C. M. Sperberg-McQueen et al.: Extensible Markup Language (XML) 1.0, W3C Recommendation. 2. vydanie. World Wide Web Consortium, 2000.
`http://www.w3.org/TR/REC-xml`

(CD99) Why Security Policies Fail. Control Data Systems, Inc. White Paper, 1999.
`www.securityfocus.com/data/library/Why_Security_Policies_Fail.pdf`

(DEL) online dokumentácia k jazykom Delphi a Kylix.
`http://bdn.borland.com/delphi/`

(FCK95) D. Ferraiolo, J. Cugini, R. Kuhn: Role-Based Access Control (RBAC): Features and Motivations. National Institute of Standards and Technology. In: Annual Computer Security Conference, IEEE Computer Society Press, 1995.

(FG99) D. Ferraiolo, S. Gavrila: A Method for Visualizing and Managing Role-Based Policies on Identity-Based Systems. Proceedings of the ACM Infosec99, Šanghaj, Čína, 1999.

(GTK) online dokumentácia ku knižnici Gimp Toolkit.
`http://www.gtk.org/`

(JAVA) online dokumentácia k jazyku Java.
`http://java.sun.com/`

(KMP00) M. Koch, L. V. Mancini, F. Parisi-Presicce: A Formal Model for Role-Based Access Control Using Graph Transformation. In: Proc. of the 6th European Symposium on Research in Computer Security, 2000, str. 122-139.

(L71) B. W. Lampson: Protection. In: Proc. 5th Princeton Conference on Information Sciences and Systems, Princeton, USA, 1971.

(LR94) C. Lewis, J. Rieman: Task-Centered User Interface Design: A Practical Introduction. Institute of Cognitive Science, University of Colorado, 1994. Online verzia: `http://hcibib.org/tcuid/`

(MYSQL) online dokumentácia k databáze MySQL
`http://www.mysql.com/doc/en/index.html`

(NS01) G. Neumann, M. Strembeck: Design and Implementation of a Flexible RBAC-Service in an Object-Oriented Scripting Language, In: Proc. of the 8th ACM Conference on Computer and Communication Security (CCS), Philadelphia, USA, 2001.

(ORACLE) online dokumentácia k databáze Oracle 9i.
`http://otn.oracle.com/pls/db92/db92.homepage`

(QT) online dokumentácia ku knižnici QT.
`http://doc.trolltech.com/3.1/`

(RWEB) dokumentácia k systému RBAC/Web; Software Diagnostic and Conformance Testing Division (SDCT), Information Technology Laboratory, NIST, USA.

(SCFY95) R. Sandhu, E. Coyne, H. Feinstein, C. Youman: Role Based Access Control Models. In: IEEE Computer, vol. 29, number 2, 1996, str. 38-47.

(SFK00) R. Sandhu, D. Ferraiolo, R. Kuhn: The NIST Model For Role-Based Access Control, Towards A Unified Standard.National Institute of Standards and Technology, USA, 2000.

(T02) R. Trebula: Vizuálny návrh bezpečnostnej politiky RBAC, diplomový projekt. Katedra informatiky a výpočtovej techniky, Fakulta elektrotechniky a informatiky STU, Bratislava, 2002.

(TCL) online dokumentácia k jazyku TCL.
`http://www.tcl.tk/doc/`

(VB) online dokumentácia k jazyku Visual Basic.
`http://msdn.microsoft.com/vbasic/`

(VV01) G. van der Veer, H. van Vliet: The Human-Computer Interface is the System. In: Proc. of the Conference on Software Engineering Education and Training, 2001, str. 276-287.

B Obsah priloženého média

Elektronické médium priložené k tomuto dokumentu obsahuje tieto súbory a adresáre:

- dp-trebula-2003.pdf – elektronická verzia tohto dokumentu,
- ACDesigner/ – binárne súbory systému ACDesigner – jadro a implementované moduly,
- sources/ – zdrojové kódy ostatnej verzie systému ACDesigner,
- javadoc/ – dokumentácia k zdrojovému kódu,
- cvs/ – archív predchádzajúcich verzií systému ACDesigner vo forme súborov systému CVS,
- biblio/ – niektoré použité literárne pramene v elektronickej forme,
- jre-1.4.1/ – adresár s inštalačnými súbormi prostredia pre beh aplikácií Java Runtime Engine pre rôzne platformy,
- mysql-j/ – ovládač pre databázu MySQL pre prostredie java.