BEZPEČNOST IS P6
2008-03-27

BEZPEČNOST V OPERAČNÍCH SYSTÉMECH:

Něco z historie:
- ✓ V počátcích vypočetni techniky byly používány kompaktní kódy přímo ovládající jednotlivá zařízeni výpočetního systému
- ✓ Prvnimi predchudci dnesnich OS byly tzv. exekutivy (executives) - jednoduche linkery, loadery, ...
- ✓ Zlom predstavoval nastup monitoru - prvnich multiprogramovych systemu, sami ridi pridelovani prostredku systemu, nastupuje problem vzajemne ochrany uzivatel

Chranene objekty:
- ✓ Pamet
- ✓ Sdilena zarizeni typu disky
- ✓ Seriove znovupouztelna zarizeni – tiskarny, pasky
- ✓ Spustitelne programy
- ✓ Sdilena data

Metody ochrany objektu v operacnich systémech:
- ✓ **Fyzicka separace** - procesy pro vykonavani operaci ruzneho stupne utajeni pouzivaji oddelena zarizeni
- ✓ **Časova separace** - procesy ruzneho stupne utajene jsou provadeny v ruznem case
- ✓ **Logicka separace** - operacni system zajistuje oddeleni jednotlivych procesu tak, ze pro kazdy vytvari iluzi, ze ma cely pocitac pro sebe
- ✓ **Kryptograficka separace** - pouzitim kryptografickych metod procesy provadeji ukryti svych dat (napr. sdileni komunikacnich linek)
- ✓ Samozrejme je mozna kombinace nekolika metod separace
- ✓ Metody jsou serazeny dle rostouci (implementacni) slozitosti a zaroven dle klesajici spolehlivosti

System muze poskytovat ruzne urovne ochrany objektu:
- ✓ **Zadna ochrana** - postacujici pokud dochazi k samovolne casove separaci
- ✓ **Isolace** - (semi)paralelne bezici procesy jsou zcela oddeleny a vubec o sobe vzajemne nevi, system zajistuje uplne ukryti objektu ostatnich procesu
- ✓ **Sdileni vseho nebo niceho** - vlastnik objektu deklaruje, zda je objekt pristupny vsem (public), nebo soukromy (private) a tedy viditelny jen pro neho
- ✓ **Sdileni s omezenymi pristupy** - OS testuje opravnenost kazdeho pokusu o pristup k danemu objektu, k danemu objektu a subjektu existuje zaznam zda subjekt ma pravo pristupovat k prislusnemu subjektu
- ✓ **Sdileni podle zpusobilosti** (share by capability) - nadstavba predchoziho zpusobu sdileni, rozsah opravneni muze dynamicky zaviset na aktualnim kontextu
- ✓ **Limitovane pouziti objektu** - nespecifikuje pouze zda subjekt smi pristupovat k danemu objektu, ale i operace, ktere subjekt smi s objektem provadet
- ✓ Seznam je opet serazen podle (implementacni) slozitosti, ktere tentokrat primo umerne odpovida kvalita poskytovane ochrany
- ✓ **Granularita** - kontrola pristupu muze byt implementovana na ruznych urovnich (byte, veta, soubor, adresar, ...), je potrebne volit mezi reziji kontroly a dostatecne jemnym rozlisenim

Ochrana pameti a adresovani:
- ✓ Ochrana pameti je zakladnim pozadavkem pro zajisteni bezpecnosti, ma-li byt spolehliva, je nutna hardwarova podpora.
- ✓ HW podpora navic poskytuje dostatecnou efektivitu ochrany.

Ohrada (fence):
- ✓ Stanovi se hranice, operacni pamet na jednu stranu od teto hranice pouziva OS, na druhou stranu aplikacni programy

- ✓ Metoda je vhodna pro jednoduche jednouzivatelske systemy, umoznuje vsak pouze chranit OS, nemuze byt pouzita pro vzajemnou ochranu uzivatelu vetsiho systemu
- ✓ Impalementace velmi jednoducha:
  - Stroj ma pevne zabudovanou tuto hranici
  - Stroj ma tzv. fence register, jehoz hodnotu porovnava s kazdou adresou, kterou aplikacni program vygeneruje

Relokace:
- ✓ Programy jsou vytvoreny tak, jako by v pameti lezely od adresy 0, v ramci procesu spousteni programu je ke vsem odkazum v programu pripocten relokacni faktor
- ✓ Aplikace tak nemuze zasahovat do oblasti, v nichz lezi system
- ✓ Metoda ma stejne nevyhody, jako predchozi zpusob ochrany pameti

Base/Bound registry:
- ✓ Preneseni myslenek predchozich metod do prostredi multiuzivatelskych systemu
- ✓ K adresam generovanym programem je pripocitavana hodnota bazoveho registru, kazdy odkaz je oprovnavan s hodnotou bound registru, zda je mensi
- ✓ Program ma tak shora i zdola omezen prostor, v nemz muze pracovat
- ✓ Metoda umoznuje vzajemne oddeleni jednotlivych uzivatelu, nechrani vsak kod aplikacniho programu pred chybou
- ✓ Moznym rozsirenim je pouzivat dva pary registru, jeden pro vymezeni oblasti pro kod procesu, druhy pro datovou zonu
- ✓ Nevyhodou je nemoznost selektivniho sdileni pouze nekterych dat

Znackovana (Tagged) architektura:
- ✓ S kazdou adresou (slovem) v pameti stroje je spojeno nekolik tag bitu, jejichz obsah urcuje typ zde ulozenych dat a povolene operace
- ✓ Obsah tag bitu je testovan pri kazdem pristupu k obsahu teto adresy, tag bity mohou byt meneny pouze privilegovanymi instrukcemi
- ✓ Alternativou muze byt pouzivani jednoho tagu pro cely blok slov

Segmentace:
- ✓ Cely program sestava z nekolika bloku - segmentu - ktere mohou byt nezavisle ulozeny do pameti
- ✓ Program potom generuje odkazy ve tvaru <jmeno_segmentu><ofset>
- ✓ Jmeno_segmentu je pomoci systemem udrzovaneho segmentoveho adresare prevedeno na adresu pocatku segmentu, ke ktere je pricten ofset
- ✓ Casto je tez ofset porovnan s velikosti segmentu, aby se zajistilo, ze program nesaha "za segment"
- ✓ Metoda jiz poskytuje dostatecne prostredky pro sdileni dat, navic umoznuje ochranu kodu programu a pripadne i vybranych dat. Rovnez je schopna chranit uzivatele navzajem.

Strankovani:
- ✓ Metoda velmi podobna segmentaci, jen predpokladame segmenty konstantni velikosti = stranky
- ✓ Opet dvouslozkove adresovani <cislo_stranky><ofset> , ochrana proti adresovani za stranku je vyresena samovolne tim, ze nezle udelat vetsi ofset, nez je velikost stranky
- ✓ Možnost ochrany obsahu stranek je ponekud slabsi nez v pripade segmentu, nebot neni prilis jasna vzajemna souvislost obsahu stranek a rozdeleni programu a dat do stranek

Ochrana obecnych objektu – s rozvojem multiprogramovani vzrusta skala objektu, ktere je treba chranit:
- ✓ Pamet
- ✓ Soubory a data na zaznamovych zarizenich
- ✓ Bezici programy
- ✓ Adresare souboru
- ✓ Hardwarova zarizeni
- ✓ Ruzne datove struktury (stack,...)
- ✓ Interni tabulky OS
- ✓ Ruzne instrukce

- ✓ Hesla a autentizacni mechanismy
- ✓ Vlastni ochranny mechanismus
- ✓ Na rozdil od problemu ochrany pameti, zde nemusi existovat centralni arbitr, pres ktereho jsou smerovany vsechny pristupy, navic typu pristupu muze byt cela rada

Cile ochrany objektu:

- ✓ **Kontrolovat kazdy pristup** - subjekt muze pozbyt pristupova prava a tedy je nutno mu zabranit v dalsim pouzivani objektu
- ✓ **Povoleni co nejmensich prav** - subjekt by mel mit pouze nejmensi mozna opravneni nutna ke korektnimu plneni jeho ukolu a to i v pripade, ze pripadna dalsi prava by pro nej byla bezcenna - toto usporadani snizuje moznost pruniku v pripade selhani casti ochranneho mechanismu
- ✓ **Overeni prijatelneho pouzivani** - nekdy je daleko podstatnejsi nez prideleni ci odepreni pristupu moci kontrolovat, co subjekt s danym objektem provadi

Mechanismy ochrany obecnych objektu:

- ✓ **Adresar (directory):**
  - Metodu popiseme pro pripad uzivatelu systemu v roli subjektu a souboru coby objektu, lze ji vsak snadno rozsirit na libovolne objekty a subjekty
  - Kazdy soubor ma sveho vlastnika, ktery k nemu vlastni veskera prava vcetne prava urcovat rozsah opravneni ostatnich uzivatelu k tomuto souboru
  - S kazdym uzivatelem je spojena specielni struktura - adresar - obsahujici odkazy na vsechny soubory, k nimz ma dany uzivatel nejake opravneni, vcetne popisu tohoto opravneni
  - Zadny uzivatel nesmi zapisovat do sveho adresare
  - Nevyhodou muze byt velky rozsah adresaru a velmi obtizna sprava a upravy takto pridelovanych opravneni. Rovnez udrzeni prehledu o tom, kdo k danemu souboru ma jaka prava muze byt problematicke.
- ✓ **Seznam opravneni (Access Control List):**
  - Opacny pristup k problemu
  - Tentokrat je s kazdym objektem udrzovan seznam informaci, ktere subjekty k nemu maji jaka opravneni
  - Metoda umoznuje snadno pridelovat implicitni prava subjektum pripadne skupinam subjektu
  - Pri vhodnem oznaceni subjektu a pouziti expanznich znaku muze byt tado metoda dostatecne pruzna
  - Pr: Pepek_Group1_Troja
    *_Group1_*
  - S eznamy zpravidla byvaji udrzovany setridene tak, ze zaznamy s expanznimi znaky jsou na konci - tak staci hledat prvni shodu s identifikaci subjektu a pouzit timto zaznamem specifikovane opravneni
- ✓ **Pristupova matice (Access Control Matrix):**
  - Radky matice odpovidaji jednotlivym subjektum, sloupce objektum
  - V policku danem radkem a sloupcem je zaznam o urovni opravneni odpovidajiciho subjektu k prislusnemu objektu
  - Pristupova matice je zpravidla velmi velka zalezitost, zhusta ridka
- ✓ **Zpusobilost (Capability):**
  - Zpusobilost budeme chapat jako nefalsovatelny token, jehoz vlastnictvi dava vlastnikovi specificka prava k danemu objektu. Lze chapat jako listek do kina.
  - Jednou z metod zajsteni nefalsovatelnosti je, ze tokeny se nepredavaji primo subjektum, ale jsou udrzovany v chranene oblasti pameti, pristupne pouze systemu
  - Pri pristupu k objektu tak system zkontroluje existenci prislusneho tokenu, tento postup lze urychlit tim, ze zvlast udrzujeme seznam Zpusobilosti prave beziciho procesu
  - Vyhodou metody je, ze dovoluje definovat nove dosu nezname zpusoby pouzivani objektu a pridelovat odpovidajici opravneni
  - Nevyhodou opet ponekud obtizna sprava techto tokenu, zejmena odebrani Zpusobilosti je netrivialni operace
- ✓ **Proceduralne orientovany pristup:**
  - Namisto pridelovani obecneho pristupu k subjektu (cteni, zapis, ...) muzeme pridelovat pravo pouzivat nekterych funkci z rozhrani, prostrednictvim ktereho je objekt zpristupnovan
  - Metoda podporuje koncept skryvani a zapouzdrovani informaci popsany v minule lekci
  - Nevyhodou je jista ztrata efektivity a rychlosti pristupu

Ochrana souboru:

- ✓ Kazdy multiuzivatelsky system musi poskytovat mechanismus na ochranu souboru
- ✓ **Opet z historie:**
  - Puvodni IBM OS poskytoval nasledujici zpusob ochrany
  - Uzivatele jsou v zasade duveryhodni a protoze nechteji, aby jim nekdo neco udelal s jejich soubory, nedelaji to ostatnim
  - Pristup ke specifickym souborum muze administrator vazat na zadani hesla
- ✓ **Ochrana po skupinach:**
  - Uzivatele jsou podle sveho zamereni, pracovniho zarazeni, ..., vhodne rozdeleny do skupin
  - Pro ucely ochrany souboru je svet rozdelen na vlastnika souboru, skupinu, do ktere vlastnik patri a ostatni uzivatele
  - Predpokada se, ze uzivatele v ramci skupiny potrebuji sdilet data
  - Pri vytvoreni souboru vlastnik specifikuje, jaka prava prideluje sobe, uzivatelum ve stejne skupine, ostatnim
  - Metoda je jednoducha, snadno implementovatelna lec neposkytujici dostatecne jemne rozliseni, navic je vetsinou nutne, aby kazdy uzivatel byl prave v jedne skupine, jinak nastavaji problemy s pridelovanim prav skupinam
- ✓ **Hesla nebo jine tokeny:**
  - Při vytvoreni souboru vlastnik specifikuje hesla, potrebna pro jiste mody pristupu k souboru, heslo zasle uzivatelum, kteri maji mit pristup
  - System splni zadost o pristup k souboru pouze tomu, kdo se prokaze odpovidajicim heslem
  - Nevyhodou je, ze v pripade zapomenuti neni mozno zjistit, jak heslo vypadalo, v pripade, ze dojde k vyzrazeni hesla je slozite nastavit nove, stejne obtizne je odejmout pravo pristupu
- ✓ **Docasne propujceni opravneni:**
  - Mechanismus znamy ze systemu UNIX.
  - Stejne pridelovani prav jako v pripade ochrany po skupinach, navic je mozno stanovit, ze (spustitelny) soubor smi byt provaden s opravnenim vlastnika
  - Prostrednictvim rutin bezicich s opravnenim vlastnika lze rizene pristupovat k souborum, ke kterym uzivatel primy pristup nema
  - Problemem popsanych schemat je jista tezkopadnost, uzivatel nemuze selektivne pridelovat prava jistym uzivatelum k jistym skupinam souboru
  - Kontrolni matice a podobne metody jsou zase prilis rozsahle a obtizne spravovatelne
- ✓ **VAX VMS/SE:**
  - Ke kazdemu souboru muze uzivatel vytvorit Seznam opravneni udavajici kdo ma jaka prava
  - Kazdy uzivatel je clenem jedne skupiny, navic administrator muze vytvorit skupinu typu obecny identifikator , a tuto skupinu mohou uzivatele uvadet v Seznamech opravneni
  - Seznamy opravneni mohou byt tez pouzity pro pridelovani pristupu k ostatnim systemovym zdrojum

Autentizace subjektu:

- ✓ Shora popsane bezpecnostni mechanismy odvijeji svoji cinnost od informace, kdo zada jaky pristup
- ✓ Je tedy nutne mit mechanismus pro autentizaci subjektu

Hesla:

- ✓ Jednoduche myslenkove i implementacne, overuje se platnost paru <identifikace:heslo>
- ✓ Mechanismus prijeti teto dvojice by nemel poskytovat utocnikovi zbytecne informace o systemu
  - Je vhodne, aby vyhodnocoval az korektnost zadani cele dvojice
  - Nekdy je proces autentizace zamerne pomaly - sekundy az desitky sekund - coz znemoznuje hadani hesel
  - Nemel by podavat informace o pricine chyby,
  - Nemel by umoznovat neomezene zkouseni
- ✓ Prihlasovani do systemu lze navic omezit na urcity cas ci misto, jisty omezeny pocet soucasnych prihlaseni daneho uzivatele do systemu
- ✓ **Hledani hesel** – system musi mit moznost kontrolovat korektnost zadaneho hesla, tedy je nutne aby udrzoval informace o vsech heslech
- ✓ **Textove soubory hesel:**

- Soubor obsahuje v textove podobe dvojice <identifikace:heslo>
- Velmi nevhodne, hesla lze zjistit z odcizenych zaloh, z dumpu pameti pri padech systemu, dojde-li chybou nektere komponenty systemu v vyzrazeni souboru hesel

✓ **Zasifrovane soubory hesel:**
- Pro sifrovani je mozne pouzit konvencnich sifer, nebo lepe kryptografickych hasovacich funkci
- Soubory zasifrovanych hesel mohou byt volne pristupne
- Protoze by v pripade dvou uzivatelu se stejnym heslem vysla stejna sifra, je vhodne pred sifrovanim k heslu pridat nahodny retezec (salt), salt je uchovavan zaroven se zasifrovanym heslem a pri verifikaci vzdy pridan k zadanemu heslu

✓ **One time passwords:**
- Resi problemy uschovy hesel
- Namisto konstantni fraze ma uzivatel prirazenu konstantni funkci (vhodny matematicky vypocet, desifrovani soukromym k.licem, ...)
- V procesu autentizace obdrzi od systemu nahodne zvolene vstupni parametry a odpovi vysledkem
- Metoda obzvlaste vhodna pro vzajemnou autentizaci stroju

Navrh bezpecnych operacnich systemu:

✓ Na kvalite operacniho systemu zavisi bezpecnost celeho mechanismu ochrany dat
✓ OS kontroluje chovani uzivatelu a programu a v konecnem dusledku zpristupnuje utajovane informace
✓ **Proces vyvoje bezpecneho OS** lze rozdelit do nekolika fazi:
- **Bezpecnostni** modely – vytvori se formalni modely prostredi a zkoumaji se zpusoby, jak v tomto prostredi zajistit bezpecnost
- **Navrh** – po zvoleni vhodneho modelu je hledan vhodny zpusob implementace
- **Overovani** – je treba overit, ze navrzena implementace skutecne odpovida teoretickemu modelu
- **Implementace** – prakticke a dukladne provedeni shora uvedenych teoretickych uvah

Bezpecnostni modely:

✓ Prvni fazi tvorby bezpecneho OS je volba vhodneho bezpecnostniho modelu
✓ Pripomenme dodrzeni zakladnich pozadavku bezpecnosti:
✓ Utajeni, integrita, dostupnost
✓ Dale budeme predpokladat, ze umime rozhodnout, zda danemu subjektu poskytnout pristup k pozadovanemu objektu, modely poskytuji pouze mechanismus pro rozhodovani

Jednourovnove modely:

✓ Jsou vhodne pouze pro pripady, kdy staci jednoduche ano/ne rozhodovani, zda danemu subjektu poskytnout pristup k pozadovanemu objektu
✓ **Monitor model**, tez reference monitor:
- Subjekt pri pristupu k objektu vyvola tzv. monitor a preda mu zadost jakou akci s kterym objektem chce provest
- Monitor zadost vyhodnoti a na zaklade informaci o pristupovych pravech vyhovi ci nikoliv
- Vyhodou jednoduchost a snadna implementovatelnost
- Nevyhodou je, ze proces poskytujici sluzby monitoruje volan pri kazdem pristupu k libovolnemu objektu, coz system velmi zatezuje
- Dalsi nevyhodou je, ze tento model je schopen kontrolovat pouze prime pristupy k datum, ale neni schopen zachytit napr. nasledujici pripad:

```
if profit <= 0 then
   delete file F
else
   write file F, "_zprava_"
endif
```

- Subjekt majici legitimni pristup k souboru F muze ziskavat informace o promenne profit, k niz by pristup mit nemel

✓ **Information flow model:**
- Odstranuje posledne jmenovanou nevyhodu predchoziho modelu
- Autori si vsimli, ze uzivatel muze ziskavat i jine informace, nez na ktere se explicitne pta
- Jiz ve fazi vyvoje je provadeno testovani vsech modulu, zda jejich vystupy zavisi na interakcich se senzitivnimi daty a pripadne jakym zpusobem

- Z techto dilcich vysledku je sestavovan celkovy graf zavilosti
- Veskere pozadavky na system prochazeji inteligentnim filtrem, ktery zjistuje, zda nedochazi k nezadouci kompromitaci informaci

Viceurovnove modely:

✓ V predchozich modelech jsme meli jednoduche vztahy objekt je/neni senzitivni, subjekt ma/nema pristup k danemu objektu

✓ Obecne vsak muze byt nekolik stupnu senzitivity a "opravnenosti"

✓ **Military security model:**

- U zelenych mozku je kazda informace zarazena do nektere z kategorii utajeni (napr. unclassified, confidental, secret, top secret), ktere jsou disjunktni
- Silne uplatneni zde ma princip nejmensich privilegii - kazdy subjekt ma mit pouze takova opravneni, aby mohl konat svoji praci
- Vsechny chranene informace jsou rozdeleny podle obsahu do oblasti (compartments), informace muze byt i nekolika oblastech zaroven
- Klasifikaci informace potom rozumime dvojici <stupen_utajeni, oblasti>
- Aby subjekt mohl pouzivat pozadovanou informaci, musi mit dostatecne opravneni.
- Opravneni ma stejny tvar jako klasifikace - <stupen_utajeni, oblasti>, tedy dany subjekt smi pouzivat informace az do stupen_utajeni v techto oblastech.

$$O \le S \Leftrightarrow st\_utaj_O \le st\_utaj_S \wedge oblast_O \subseteq oblast_S$$

- Relace $\le$ odpovida opravneni subjektu S k danemu objektu O.
- Pozadavky na stupen utajeni byvaji oznacovany jako hierarchicke, rozdeleni na oblasti jako nehierarchicke omezeni.

✓ **Svazovy model (Lattice model):**

- Predchozi military model je specielnim pripadem tohoto modelu
- Relace $\le$ je castecnym usporadanim, mnozina klasifikaci vsech informaci v systemu tvori svaz, stejne tak mnozina opravneni vsech subjektu
- V runych oblastech se pouziva ruznych svazu, napr. v komercni oblasti jsou obvykle stupne utajeni public, company confidental, high security, rovnez rozdeleni do oblasti se lisi pripad od pripadu...
- Svazovy model je casto pouzivanym modelem v mnoha prostredich
- Dale popiseme dva modely, zabyvajici se tokem informaci uvnitr systemu

✓ **Bell-LaPadula model:**

- Model popisuje povolene presuny informaci, takove, aby bylo zajisteno jejich utajeni
- Pro kazdy subjekt S resp. objekt O v systemu necht je definovana bezpecnostni trida C(S) resp. C(O)
- Bezpecne presuny informaci maji nasledujici vlastnoti:
- Vlastnost jednoduche bezpecnosti (Simple Security Property):
- Subjekt S muze cist objekt O prave kdyz $C(O) \le C(S)$
  -vlastnost (-Property):
- Subjekt S majici pravo cteni k objektu O muze zapisovat do objektu P prave kdyz $C(O) \le C(P)$
- Obycejne nepotrebujeme tak silna omezeni, ktera klade *-vlastnost. Casto je tato vlastnost ponekud oslabena v tom smyslu, ze system povoli zapis do objektu nizsi bezpecnostni tridy, pokud zapisovana data nezavisi na ctenych udajich.
- Model byl je pouzivan v systemech, ktere paralelne zpracovavaji informace ruzneho stupne utajeni.

✓ **Biba model:**

- predchozi model se vsak vubec nezabyva integritou dat, Biba model je dualnim modelem k Bell-LaPadula modelu
- Necht pro kazdy subjekt S resp. objekt O v systemu je definovana integritni bezpecnostni trida I(S) resp. I(O). Obdobne jako v predchozim pripade definujeme:
- Vlastnost jednoduche integrity (Simple Integrity Property):
- Subjekt S muze modifikovat objekt O prave kdyz $I(O) \le I(S)$
  Integritni *-vlastnost (Integrity *-Property):
- Subjekt S majici pravo cteni k objektu O muze zapisovat do objektu P prave kdyz $I(O) \ge I(P)$

- Biba model se zabyva zajistenim integrity a tedy i duveryhodnosti dat. Bepecnostni trida entity vpodstate popisuje miru jeji duveryhodnosti pro ostatni.
- Tento model vubec neresi utajeni dat.
- Prestoze byla ucinena rada pokusu o nalezeni kompromisu mezi zajistenim integrity a utajeni, dosud neexistuje obecne prijaty model, ktery by resil oba problemy. Nasledujici modely se zabyvaji teoretickymi limity abstraktnich bezpecnostnich systemu.

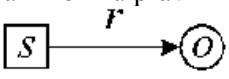
✓ **Graham-Denning model**

- Model pracuje s mnozinou subjektu S, mnozinou objektu O, mnozinou prav R a pristupovou matici A.
- Kazdy objekt ma prirazen jeden subjekt nazyvany vlastnik, kazdy subjekt ma prirazen jiny subjekt nazyvany kontroler.
- Model definuje nasledujici prava:
  - Vytvorit objekt – povoluje subjektu vytvorit v systemu novy objekt
  - Vytvorit subjekt, zrusit objekt, rusit subjekt – obdobne jako predchozi
  - Cist pristupova prava – povoluje subjektu zjistit aktualni pristupova prava jisteho subjektu k urcitemu subjektu
  - Pridelit pristupova prava – dovoluje vlatniku objektu pridelit jista prava k objektu urcitemu subjektu
  - Zrusit pristupova prava – dovoluje vlastniku objektu resp. kontroleru subjektu odebrat danemu subjektu jista prava k objektu resp. subjektu
  - Predat pristupova prava – dovoluje subjektu predat nektere ze svych prav jinemu subjektu (kazde opravneni muze byt predatelne ci nikoliv, obdrzi-li subjekt predatelne pravo, muze jej dale predat jako predatelne ci nepredatelne).
- Nasledujici tabulka uvadi podminky nutne pro vykonani operaci s pristupovymi pravy.

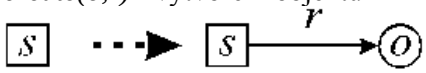
vytvorit objekt *o*	-
vytvorit subjekt *s*	-
zrusit objekt *o*	vlastnik je v $A[x,o]$
zrusit subjekt *s*	vlastnik je v $A[x,s]$
cist pristupova prava *s* k *o*	kontroler je v $A[x,s]$, nebo vlastnik v $A[x,o]$
zrusit pristupove pravo *r* subjektu *s* k *o*	kontroler je v $A[x,s]$, nebo vlastnik v $A[x,o]$
pridelit *s* pravo *r* k objektu *o*	vlastnik je v $A[x,o]$
predat pristupove pravo *r* nebo *r** k objektu *o* subjektu *s*	*r** je v $A[x,o]$

*r** oznacuje predatelne pravo

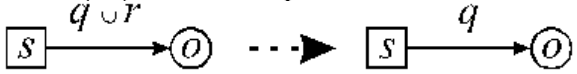
✓ **Take-Grant systém:**

- Model pracuje s ctyrmi zakladnimi primitivami: create, revoke, take, grant.
- Predpokladame, ze systemu obsahuje mnozinu subjektu S, mnozinu objektu O, objekty delime na aktivni (zaroven i subjekty) a pasivni (nejsou subjekty) a mnozinu prav R
- Pro popis operaci pouzijeme nasledujici notaci: 
- Subjekt s ma k objektu o opravneni r.

create(o,r) - vytvoreni objektu



revoke(o,r) - odebrani opravneni



grant(o,p,r) - predani opravneni

take(o,p,r) - prevzeti opravneni

- Vyhodou popsaneho systemu je, ze umoznuje v subpolynomialnim case resit dotazy na dostupnost jisteho objektu pro dany subjekt.

Navrh bezpecneho operacniho systemu:

✓ **Implementace** bezpecnostnich mechanismu je v primem rozporu s efektivitou systemu

✓ OS vykonava nekolik s bezpecnostni uzce souvisejicich cinnosti:
- Autentizace uzivatelu
- Ochrana pameti - mezi uzivateli i v ramci jednoho uzivatelskeho prostoru
- Rizeni pristupu k souborum a I/O zarizenim - ochrana pred neautorizovanym pristupem
- Alokace a rizeni pristupu k obecnym objektum - zajisteni bezproblemoveho soucasneho pristupu vice uzivatelu k stejnemu objektu
- Zabezpeceni sdileni - zejmena zajisteni integrity a konzistentnosti
- Zajisteni spravedliveho pristupu - o HW prostredky se opirajici mechanismus zajistujici, ze vsichni uzivatele dostavaji pridelen procesor a ostatni systemove zdroje
- Meziprocesova komunikace a synchronizace - system poskytuje mechanismus pro bezpecne predavani zprav mezi procesy, procesy nekomunikuji primo ale via system

✓ **Bezpecnost** musi byt brana v potaz ve vsech aspektech navrhu systemu a musi byt zapracovana jiz v prvotnim navrhu. Je velmi obtizne ji "pridat" do hotoveho navrhu.

✓ Nasledujici **principy** je vhodne mit na pameti:
- **Nejmensi prava** – kazdy subjekt by mel mit pouze nezbytna prava
- **Ekonomicky navrh** – bezpecnosni system ma byt maly a jednoduchy, pak je testovatelny a verohodny
- **Otevreny navrh** – bezpecnostni mechanismus by mel byt verejne znamy (a oponovany) a mel by zaviset na bezpecnosti co nejmene objektu
- **Uplne zprostredkovani** – veskere pristupy k objektum zprostredkovava a testuje OS
- **Povolovani operace** – co neni vyslovne povoleno, je zakazano
- **Rozdelene opravneni** – pristup k objektum by mel zaviset na vice podminkach (napr. spravna autentizace a vlastnictvi klice)
- **Nejmensi sdilene prostredky** – sdilene moduly jsou potencialnim kanalem pro unik informaci, melo by jich byt co nejmene
- **Snadna pouzitelnost** – mechanikmus neni obchazen, kdyz se neplete

✓ **Virtualni adresni prostor:**
- IBM MVS provadi logicke oddeleni uzivatelu, ktere poskytuje dojem fyzicke separace
- Pomoci mechanismu strankovani jsou cela oddeleny adresni prostory jednotlivych uzivatelu, kazdy uzivatel vidi pouze svuj prostor, do kazdeho z uzivatelskych prostoru je mappvana oblast pameti obsahujici vlastni system, cimz vznika dojem, ze uzivatel ma cely system sam pro sebe

✓ **Virtual machine:**
- operacni system IBM VM poskytuje nejen virtualni pamet, ale provadi virtualizaci celeho pocitace - I/O zarizeni, file-systemu a dalsich zdroju
- simulovany stroj muze mit naprosto odlisne vlastnosti od vlastnosti pocitace, na kterem OS VM bezi
- poskytovana ochrana je tedy daleko silnejsi
- VM byl navrhovan jako system, umoznujici na jednom fyzickem pocitaci provozovat zaroven nekolik ruznych operacnich systemu
- tak predstavuje dalsi vrstvu ochrany, nebot pokud se uzivateli podari proniknout ochrannymi mechanismy operacniho systemu, ktery pouziva, ziska pristup pouze k teto jedine domene, nebot VM mu zabrani v pristupu k celemu pocitaci

✓ **Kernel:**
- Je to cast OS provadejici nejzakladnejsi funkce - standardne synchronizace, meziprocesova komunikace, zasilani zprav a obsluha preruseni
- Tzv. **security kernel** poskytuje zaklad pro vybudovani bezpecnostniho mechanismu, casto byva implementovan uvnitr kernelu
- Uzavrit bezpecnostni funkce systemu do security kernelu ma nekolik **duvodu**:
  - Oddeleni od zbytku systemu zjednodusuje ochranu mechanismu
  - Vsechny bezpecnostni funkce jsou shromazdeny v jednom kusu kodu, tedy implementace bepecnosti je kompaktni
  - Kernel nebyva velky, tedy implementace je snadno overitelna
  - Je snazsi provadet testovani a zmeny bepecnostniho mechanismu

- Pres kernel prochazeji veskere zadosti o pristup ke vsem objektum (volani odpovidajicich modulu), tedy je mozno zachytit kazdy pristup
- **Kernel hlida zejmena:**
  - Aktivaci procesu - zajistovani context switchingu, realokaci pameti, access kontrol listu, ...
  - Stridani domen - procesy casto provadeji volani procesu bezicich v jine bezpecnostni domene, za ucelem ziskani senzitivnich informaci
  - Ochrana pameti - je nutne hlidat vsechny odkazy na pamet, aby nedochazelo k naruseni bezpecnostnich domen
  - I/O operace

✓ **Vrstvovy model (Layered design):**
- Jiz operacni systemy s kernelem obsahuji nekolik vrstev - hardware, kernel, zbytek OS, uziv. procesy
- Tyto vrstvy lze dale delit - napr. na uzivatelske urovni muzeme oddelit semi-sytemove programy jako ruzne databazove systemy, shelly apod.
- Vrstvy le chapat jako soustredne kruhy, cim blize je vrstva stredu, tim je duveryhodejsi a bezpecnejsi
- Ne vsechny bepecnostni funkce (napr. autentizace uzivatele) jsou implementovany uvnitr bezpecnostniho jadra
- Bezpecnostni jadro spolupracuje s okolnimi spolehlivymi vrstvami, ktere by meli byt formalne overeny a prinejmensim dobre otestovany
- Kazda vrstva pouziva sluzby nizsich vrstev a sama vyssim vrstvam poskytuje sluzby jiste urovne bepecnosti, stjna funkce muze byt implementovana v nekolika vrstvach zaroven

✓ **Kruhova struktura (Ring structured):**
- Kruhy cislovany od 0 (kernel), cim duveryhodnejsi proces, tim nizsi cislo kruhu, do ktereho patri
- Kruhy jsou soustredne a prekryvajici se - proces patri do kruhu k a vsech dalsich, ve stredu je HW pocitace
- Kazda procedura, nebo oblast obsahujici data se nazyva segment
- Ochrana segmentu zalozena na trojici $\langle b_1, b_2, b_3 \rangle$, $b_1 \leq b_2 < b_3$, nazyvane zavora kruhu (ring bracket), (b1, b2) nazyvame pristupova zavora (access bracket), (b2, b3) potom zavora volani (gate extension, call bracket)
- Necht programova rutina patri do kruhu k, pokud k = b1, muze pracovat primo s daty tohoto segmentu, pokud $b_1 < k \leq b_2$, muze pracovat primo s kopii dat a pokud $b_2 < k \leq b_3$, muze k datum pristupovat pouze prostrednictvim definovaneho rozhrani (gate)
- Tento zakladni mechanismus, nazyvany tez nondiscretionary nebo mandatory control muze byt dale doplnen o dalsi doplnkove (discretionary) mechanismy -
- napr. k danym datum smeji pristupovat pouze jmenovite procesy, procesy patrici do okruhu pristupove zavory mohou volne cist, ale zapisovat pouze za specifickych podminek apod.

Průniky operačním systémem

✓ místem největšího počtu průniku je mechanismus zpracování I/O operaci:
- Mnohá I/O zařízení jsou do značné míry inteligentní a nezávislá na zbytku systému, provádějí optimalizaci své činnosti, jejich řadiče často spravují více takovýchto zařízení
- Kód I/O operací je často velmi rozsáhlý, je těžké jej řádně testovat, někdy je dokonce nutné používat kód dodaný výrobcem zařízeni ...
- V zájmu rychlosti a efektivity I/O operace občas obcházejí bezpečnostní mechanismy operačního systému, jako stránkováni, segmentaci apod.
- Velká část I/O operaci je znakově orientována, v zajmu efektivity se často příslušné kontroly neprovádějí s každým přijatým znakem, ale pouze při startu operace

✓ Druhým problémem je hledání kompromisu mezi důkladnou izolací uživatelů a nutností umožnit sdílení dat, tento kompromis zhusta bývá obtížně formalizovatelný, nejasnosti návrhu pak mohou byt příčinou "děr" v implementaci

✓ Ne vždy je možné provádět kontroly oprávněnosti s každou operací, často je kontrola prováděna pouze jednou během prováděni celého bloku akcí, pokud se v této době uživateli podaří změnit parametry, může dojít k průniku

✓ Další skulinu v bezpečnosti může způsobit snaha o obecnost možného nasazení systému - aby bylo možno systém používat pro nejrůznější úkoly, ponechají návrháři často mechanismus, pomoci kterého si uživatel může systém přizpůsobit