# Bezpečnost v operačních systémech

## Něco z historie

v počátcích výpočetní techniky byly používány kompaktní kódy přímo ovládající jednotlivá zařízení výpočetního systému
prvnímy předchůdci dnešních OS byly tzv. *exekutivy* (executives) - jednoduché linkery, loadery, …
zlom představoval nástup *monitorů* - prvních multiprogramových systémů, sami řídí přidělování prostředků systému, nastupuje problém vzájemné ochrany uživatel

## *Chráněné objekty*

1. paměť
2. procesor
3. spustitelné programy
4. sdílená zařízení typu disky
5. seriově znovupoužitelná zařízení - tiskárny, pásky
6. sdílená data

## *Poskytované služby*

- autentizace
- autorizace
- řízení přístupu/logická separace
- auditovatelnost
- dostupnost
- zotavení

## *Metody ochrany objektů v operačních systémech*

- fyzická separace - procesy pro vykonávání operací různého stupně utajení používají oddělená zařízení
- časová separace - procesy různého stupně utajené jsou prováděny v různém čase
- **logická separace** - operační systém zajišťuje oddělení jednotlivých procesů tak, že pro každý vytváří iluzi, že má celý počítač pro sebe
- kryptografická separace - použitím kryptografických metod procesy provádějí ukrytí svých dat (např. sdílení komunikačních linek)

samozřejmě je možná kombinace několika metod separace
metody jsou seřazeny dle rostoucí (implementační) složitosti a zároveň dle klesající spolehlivosti

## *Ochrana paměti a adresování*

Ochrana paměti je základním požadavkem pro zajištění bezpečnosti, má-li být spolehlivá, je nutná hardwarová podpora.
HW podpora navíc poskytuje dostatečnou efektivitu ochrany.

### Ohrada (fence)

stanoví se hranice, operační paměť na jednu stranu od této hranice používá OS, na druhou stranu aplikační programy
metoda je vhodná pro jednoduché jednouživatelské systémy, umožňuje však pouze chránit OS, nemůže být použita pro vzájemnou ochranu uživatelů většího systému
implementace velmi jednoduchá:

- stroj má pevně zabudovanou tuto hranici
- stroj má tzv. *fence register*, jehož hodnotu porovnává s každou adresou, kterou aplikační program vygeneruje

### Relokace

programy jsou vytvořeny tak, jako by v paměti ležely od adresy 0, v rámci procesu spouštění programu je ke všem odkazům v programu připočten relokační faktor
aplikace tak nemůže zasahovat do oblastí, v nichž leží systém
metoda má stejné nevýhody, jako předchozí způsob ochrany paměti

### Base/Bound registry

… přenesení myšlenek předchozích metod do prostředí multiuživatelských systémů
k adresám generovaným programem je připočítávána hodnota bázového registru, každý odkaz je oprovnáván s hodnotou bound registru, zda je menší
program má tak shora i zdola omezen prostor, v němž může pracovat
metoda umožňuje vzájemné oddělení jednotlivých uživatelů, nechrání však kód aplikačního programu před chybou
možným rozšířením je používat dva páry registrů, jeden pro vymezení oblasti pro kód procesu, druhý pro datovou zónu
nevýhodou je nemožnost selektivního sdílení pouze některých dat

## Značkovaná (Tagged) architektura

s každou adresou (slovem) v paměti stroje je spojeno několik tag bitů, jejichž obsah určuje typ zde uložených dat a povolené operace
obsah tag bitů je testován při každém přístupu k obsahu této adresy, tag bity mohou být měněny pouze privilegovanými instrukcemi
alternativou může být používání jednoho tagu pro celý blok slov

## Segmentace

celý program sestává z několika bloků - segmentů - které mohou být nezávisle uloženy do paměti
program potom generuje odkazy ve tvaru <*jméno_segmentu*><*ofset*>
*jméno_segmentu* je pomocí systémem udržovaného segmentového adresáře převedeno na adresu počátku segmentu, ke které je přičten ofset
často je též ofset porovnán s velikostí segmentu, aby se zajistilo, že program nesahá "za segment"
Metoda již poskytuje dostatečné prostředky pro sdílení dat, navíc umožňuje ochranu kódu programu a případně i vybraných dat. Rovněž je schopna chránit uživatele navzájem.

## Stránkování

metoda velmi podobná segmentaci, jen předpokládáme segmenty konstantní velikosti = *stránky*
opět dvousložkové adresování <číslo_stránky><offset> , ochrana proti adresování za stránku je vyřešena samovolně tím, že nezle udělat větší ofset, než je velikost stránky
možnost ochrany obsahu stránek je poněkud slabší než v případě segmentů, neboť není příliš jasná vzájemná souvislost obsahů stránek a rozdělení programu a dat do stránek

# *Ochrana procesoru*

- privilegované instrukce – přístup ke klíčovým operacím (ovládání periferií a dalších autonomních zařízení, nastavení bezpečnostního mechanismu, přepínání úrovní, správa procesů, ovládání přerušení, ...)
- úrovně oprávnění procesu – rozdělení běžících procesů do různých tříd s diferencovanými oprávněními, přístupy k privilegovaným instrukcím, omezení přístupu k paměti, ....
- správa procesorového času

- správa využívání systémových prostředků – využití autonomních subsystémů, alokace, ...

## *Ochrana obecných objektů*

s rozvojem multi-užívání vzrůstá škála objektů, které je třeba chránit

- soubory a data na záznamových zařízeních
- běžící programy
- adresáře souborů
- hardwarová zařízení
- různé datové struktury (stack,…)
- interní tabulky OS
- hesla a autentizační mechanismy
- vlastní ochranný mechanismus

na rozdíl od problému ochrany paměti, zde nemusí existovat centrální arbitr, přes kterého jsou směrovány všechny přístupy, navíc typů přístupů může být celá řada

**cíle ochrany objektů**
*Kontrolovat každý přístup* - subjekt může pozbýt přístupová práva a tedy je nutno mu zabránit v dalším používání objektu
*Povolení co nejmenších práv* - subjekt by měl mít pouze nejmenší možná oprávnění nutná ke korektnímu plnění jeho úkolu a to i v případě, že případná další práva by pro něj byla bezcenná - toto uspořádání snižuje možnost průniku v případě selhání části ochranného mechanismu
*Ověření přijatelného používání* - někdy je daleko podstatnější než přidělení či odepření přístupu moci kontrolovat, co subjekt s daným objektem provádí

## Autentizace subjektů

shora popsané bezpečnostní mechanismy odvíjejí svoji činnost od informace, ***kdo*** žádá jaký přístup
je tedy nutné mít mechanismus pro autentizaci subjektů

## Hesla

jednoduché myšlenkově i implementačně, ověřuje se platnost páru

<identifikace:heslo>

mechanismus přijetí této dvojice by neměl poskytovat útočníkovi zbytečné informace o systému

- je vhodné, aby vyhodnocoval až korektnost zadání celé dvojice

- někdy je proces autentizace záměrně pomalý - sekundy až desítky sekund - což znemožňuje hádání hesel
- neměl by podávat informace o příčině chyby,
- neměl by umožňovat neomezené zkoušení

přihlašování do systému lze navíc omezit na určitý čas či místo, jistý omezený počet současných přihlášení daného uživatele do systému

### Hledání hesel

Systém musí mít možnost kontrolovat korektnost zadaného hesla, tedy je nutné aby udržoval informace o všech heslech

#### *Textové soubory hesel*

soubor obsahuje v textové podobě dvojice <identifikace:heslo>
velmi nevhodné, hesla lze zjistit z odcizených záloh, z dumpů paměti při pádech systému, dojde-li chybou některé komponenty systému v vyzrazení souboru hesel

#### *Zašifrované soubory hesel*

pro šifrování je možné použít konvenčních šifer, nebo lépe kryptografických hašovacích funkcí
soubory zašifrovaných hesel mohou být volně přístupné
protože by v případě dvou uživatelů se stejným heslem vyšla stejná šifra, je vhodné před šifrováním k heslu přidat náhodný řetězec (salt), salt je uchováván zároveň se zašifrovaným heslem a při verifikaci vždy přidán k zadanému heslu

#### *One time passwords*

řeší problémy úschovy hesel
namísto konstantní fráze má uživatel přiřazenu konstantní funkci (vhodný matematický výpočet, dešifrování soukromým k.líčem, …)
v procesu autentizace obdrží od systému náhodně zvolené vstupní parametry a odpoví výsledkem
metoda obzvláště vhodná pro vzájemnou autentizaci strojů

## Návrh bezpečných operačních systémů

na kvalitě operačního systému závisí bezpečnost celého mechanismu ochrany dat
OS kontroluje chování uživatelů a programů a v konečném důsledku zpřístupňuje utajované informace
proces vývoje bezpečného OS lze rozdělit do několika fází

- *bezpečnostní modely* - vytvoří se formální modely prostředí a zkoumají se způsoby, jak v tomto prostředí zajistit bezpečnost
- *návrh* - po zvolení vhodného modelu je hledán vhodný způsob implementace
- *ověřování* - je třeba ověřit, že navržená implementace skutečně odpovídá teoretickému modelu
- *implementace* - praktické a důkladné provedení shora uvedených teoretických úvah

## Návrh bezpečného operačního systému

implementace bezpečnostních mechanismů je v přímém rozporu s efektivitou systému

OS vykonává několik s bezpečnostní úzce souvisejících činností:

- autentizace uživatelů
- ochrana paměti - mezi uživateli i v rámci jednoho uživatelského prostoru
- řízení přístupu k souborům a I/O zařízením - ochrana před neautorizovaným přístupem
- alokace a řízení přístupu k obecným objektům - zajištění bezproblémového současného přístupu více uživatelů k stejnému objektu
- zabezpečení sdílení - zejména zajištění integrity a konzistentnosti
- zajištění spravedlivého přístupu - o HW prostředky se opírající mechanismus zajišťující, že všichni uživatelé dostávají přidělen procesor a ostatní systémové zdroje
- meziprocesová komunikace a synchronizace - systém poskytuje mechanismus pro bezpečné předávání zpráv mezi procesy, procesy nekomunikují přímo ale via systém

Bezpečnost musí být brána v potaz ve všech aspektech návrhu systému a musí být zapracována již v prvotním návrhu. Je velmi obtížné ji "přidat" do hotového návrhu.

Následující principy je vhodné mít na paměti:

Nejmenší práva - každý subjekt by měl mít pouze nezbytná práva

Ekonomický návrh - bezpečnosní systém má být malý a jednoduchý, pak je testovatelný a věrohodný

Otevřený návrh - bezpečnostní mechanismus by měl být veřejně známý (a oponovaný) a měl by záviset na bezpečnosti co nejméně objektů

Úplné zprostředkování - veškeré přístupy k objektům zprostředkovává a testuje OS

Povolování operace - co není výslovně povoleno, je zakázáno
Rozdělené oprávnění - přístup k objektům by měl záviset na více podmínkách (např. správná autentizace a vlastnictví klíče)
Nejmenší sdílené prostředky - sdílené moduly jsou potenciálním kanálem pro únik informací, mělo by jich být co nejméně
Snadná použitelnost - mechanismus není obcházen, když se neplete více, než je únosné

## Virtuální adresní prostor

IBM MVS provádí logické oddělení uživatelů, které poskytuje dojem fyzické separace
pomocí mechanismu stránkování jsou cela odděleny adresní prostory jednotlivých uživatelů, každý uživatel vidí pouze svůj prostor, do každého z uživatelských prostorů je mappvána oblast paměti obsahující vlastní systém, čímž vzniká dojem, že uživatel má celý systém sám pro sebe

## Virtual machine

operační systém IBM VM poskytuje nejen virtuální pamět, ale provádí virtualizaci celého počítače - I/O zařízení, file-systému a dalších zdrojů
simulovaný stroj může mít naprosto odlišné vlastnosti od vlastností počítače, na kterém OS VM běží
poskytovaná ochrana je tedy daleko silnější
VM byl navrhován jako systém, umožňující na jednom fyzickém počítači provozovat zároveň několik různých operačních systémů
tak představuje další vrstvu ochrany, neboť pokud se uživateli podaří proniknout ochrannými mechanismy operačního systému, který používá, získá přístup pouze k této jediné doméně, neboť VM mu zabrání v přístupu k celému počítači

## Kernel

… část OS provádějící nejzákladnější funkce - standardně synchronizace, meziprocesová komunikace, zasílání zpráv a obsluha přerušení
tzv. *security kernel* poskytuje základ pro vybudování bezpečnostního mechanismu, často bývá implementován uvnitř kernelu
uzavřít bezpečnostní funkce systému do security kernelu má několik důvodů:

- oddělení od zbytku systému zjednodušuje ochranu mechanismu
- všechny bezpečnostní funkce jsou shromážděny v jednom kusu kódu, tedy implementace bezpečnosti je kompaktní

- kernel nebývá velký, tedy implementace je snadno ověřitelná
- je snazší provádět testování a změny bepečnostního mechanismu
- přes kernel procházejí veškeré žádosti o přístup ke všem objektům (volání odpovídajících modulů), tedy je možno zachytit každý přístup

kernel hlídá zejména:

- aktivaci procesů - zajišťování context switchingu, realokací paměti, access kontrol listů, …
- střídání domén - procesy často provádějí volání procesů běžících v jiné bezpečnostní doméně, za účelem získání senzitivních informací
- ochrana paměti - je nutné hlídat všechny odkazy na paměť, aby nedocházelo k narušení bezpečnostních domén
- I/O operace

## Vrstvový model (Layered design)

již operační systémy s kernelem obsahují několik vrstev - hardware, kernel, zbytek OS, uživ. procesy

tyto vrstvy lze dále dělit - např. na uživatelské úrovni můžeme oddělit semi-sytémové programy jako různé databázové systémy, shelly apod.

vrstvy le chápat jako soustředné kruhy, čím blíže je vrstva středu, tím je důvěryhodější a bezpečnější

ne všechny bezpečnostní funkce (např. autentizace uživatele) jsou implementovány uvnitř bezpečnostního jádra

bezpečnostní jádro spolupracuje s okolními spolehlivými vrstvami, které by měli být formálně ověřeny a přinejmenším dobře otestovány

každá vrstva používá služby nižších vrstev a sama vyšším vrstvám poskytuje služby jisté úrovně bezpečnosti, stejna funkce může být implementována v několika vrstvách zároveň

## Kruhová struktura (Ring structured)

kruhy číslovány od 0 (kernel), čím důvěryhodnější proces, tím nižší číslo kruhu, do kterého patří

kruhy jsou soustředné a překrývající se - proces patří do kruhu *k* a všech dalších, ve středu je HW počítače

každá procedura, nebo oblast obsahující data se nazývá *segment*

ochrana segmentu založena na trojici $\langle b_1, b_2, b_3 \rangle$, $b_1 \leq b_2 < b_3$, nazývané závora kruhu (ring bracket), ($b_1$, $b_2$) nazýváme přístupová závora (access bracket), ($b_2$, $b_3$) potom závora volání (gate extension, call bracket)

nechť programová rutina patří do kruhu $k$, pokud $k = b_1$, může pracovat přímo s daty tohoto segmentu, pokud $b_1 < k \leq b_2$, může pracovat přímo s kopií dat a pokud $b_2 < k \leq b_3$, může k datům přistupovat pouze prostřednictvím definovaného rozhraní (gate)
tento základní mechanismus, nazývaný též nondiscretionary nebo mandatory control může být dále doplněn o další doplňkové (discretionary) mechanismy -
např. k daným datům smějí přistupovat pouze jmenovité procesy, procesy patřící do okruhu přístupové závory mohou volně číst, ale zapisovat pouze za specifických podmínek apod.

## *Průniky operačním systémem*

1. místem největšího počtu průniků je mechanismus zpracování I/O operací
   - 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 zařízení …
   - v zájmu rychlosti a efektivity I/O operace občas obcházejí bezpečnostní mechanismy operačního systému, jako stránkování, segmentaci apod.
   - velká část I/O operací je znakově orientovaná, v zájmu efektivity se často příslušné kontroly neprovádějí s každým přijatým znakem, ale pouze při startu operace
2. HW současných procesorů poskytuje rozsáhlé prostředky pro vytvoření kvalitní bezpečnosti (úrovně oprávnění, privilegované instrukce, trasovací režimy, systém obsluhy výjimek, systém ochrany segmentů a stránek, ...), ne vždy je však využíván
3. dalším problémem je hledání kompromisu mezi důkladnou izolací uživatelů a nutností umožnit sdílení dat, tento kompromis zhusta bývá obtížně formalizovatelný, nejasnosti návrhu pak mohou být příčinou "děr" v implementaci
4. ne vždy je možné provádět kontroly oprávněnosti s každou operací, často je kontrola prováděna pouze jednou během provádění celého bloku akcí, pokud se v této době uživateli podaří změnit parametry, může dojít k průniku
5. další skulinu v bezpečnosti může způsobit snaha o obecnost možného nasazení systému - aby bylo možno systém používat pro nejrůznější úkoly, ponechají návrháři často mechanismus, pomocí kterého si uživatel může systém přizpůsobit
   tento mechanismus ovšem může být zneužit

6. klasickým místem průniku se stávají programy, běžící v rámci OS s rozsáhlými oprávněními
7. vzhledem k současné "prosíťovanosti" světa je častým místem útoku právě obsluha komunikačního protokolu