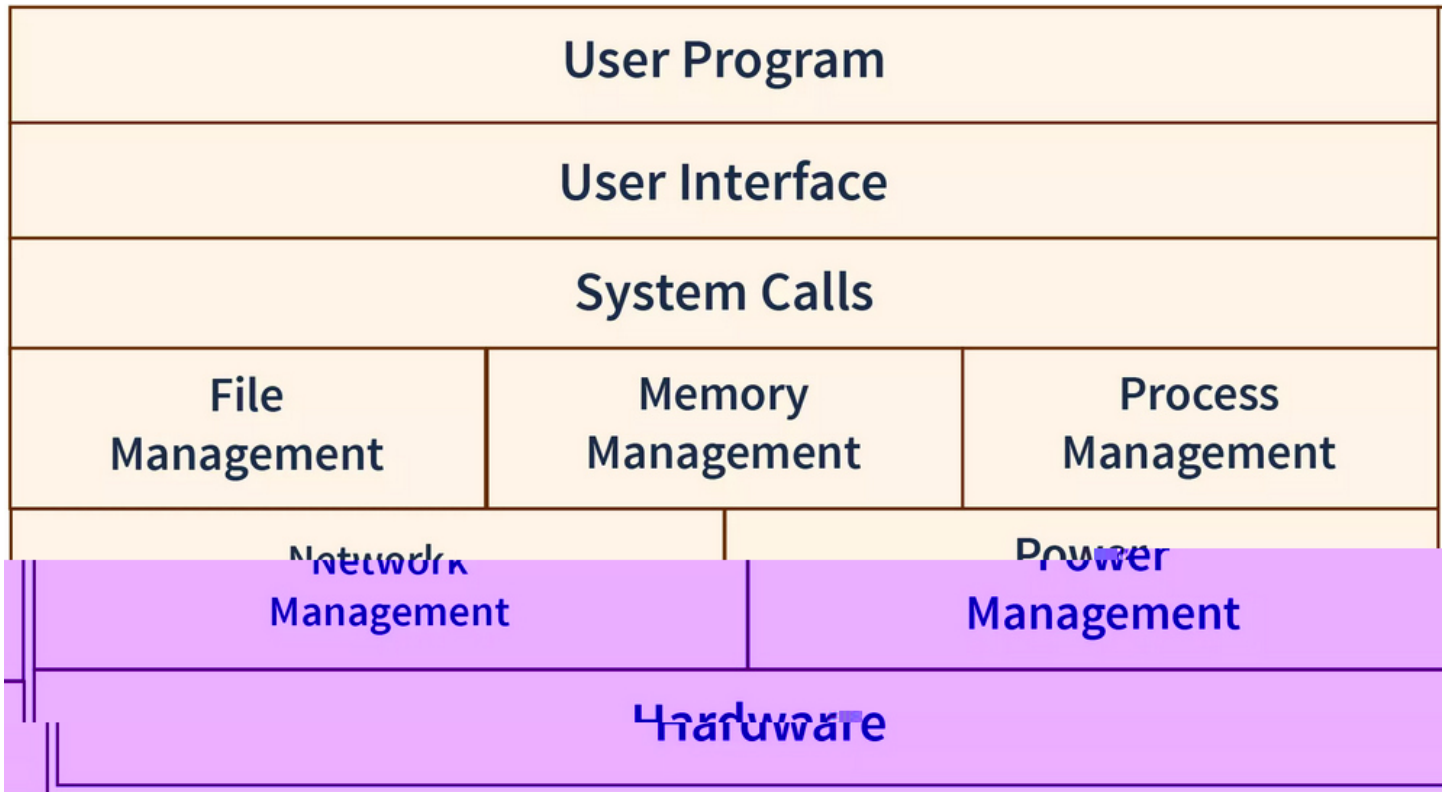
# Bezpečnostní mechanismy

## Něco ke struktuře

možný pohled na OS:




cílem OS je vytvořit virtualizaci HW zdrojů a v oblasti bezpečnosti:

- zřídit systém identit
- zajistit separaci a řízení přístupu
- možnosti obnovení po chybě
- vynutit spravedlivé využívání zdrojů

Z hlediska implementace je prostředky možné rozdělit na ty, jejichž použití

- systém zprostředkovává (procesor, paměť)
- sám vytváří (souborový systém, provozní záznamy, komunikační protokoly)

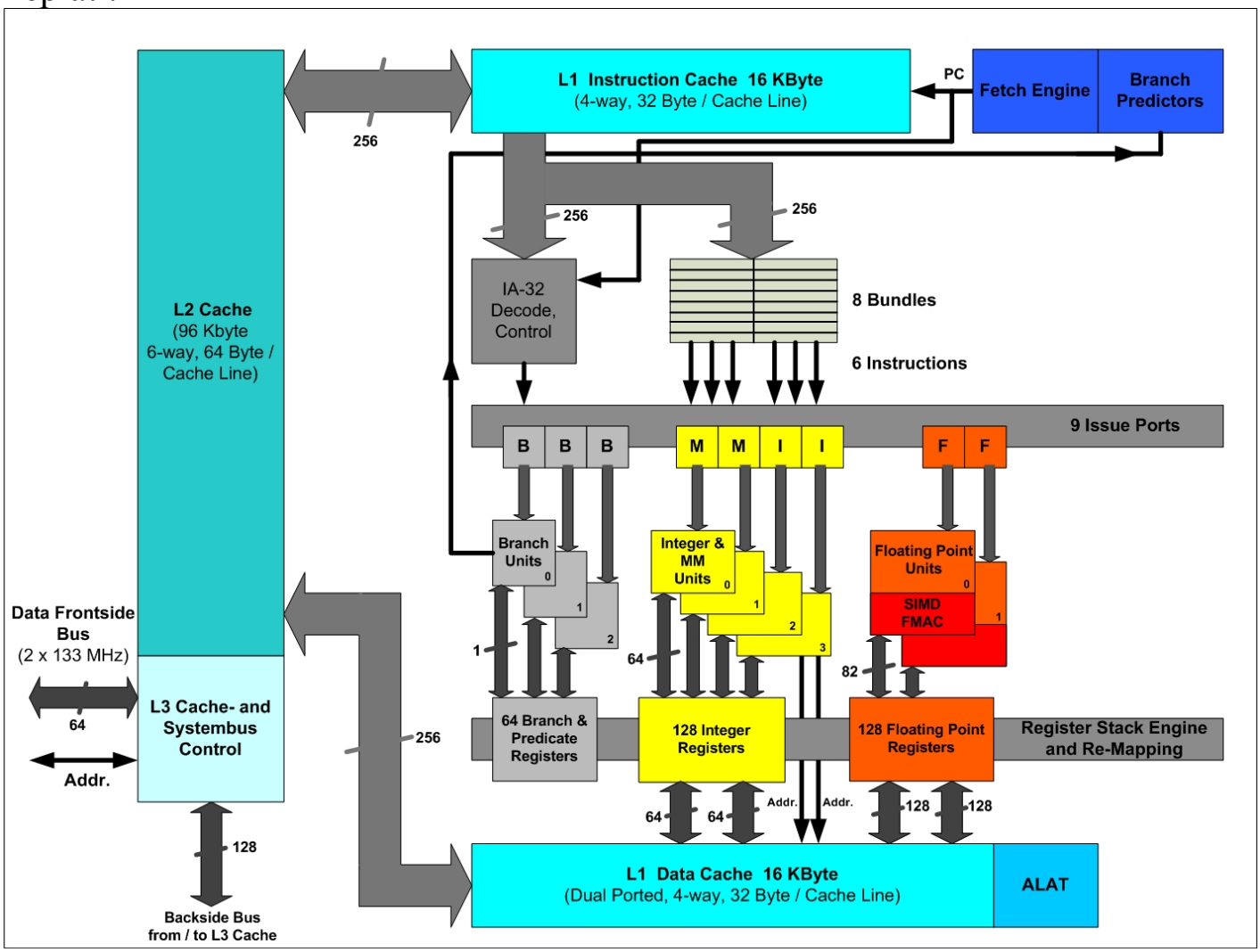
## *Typické útoky a incidenty*

- chyby HW a závady prostředí
- malware
- DoS útoky
- uživatelé

  - nedbalí autorizovaní uživatelé
  - zlomyslní autorizovaní uživatelé
  - napodobovatelé / podloudníci - externisté, kteří dokáží napodobit autorizovaného uživatele / administrátora
- buffer overflow
- portscanning

## *Ochrana procesoru*

O procesoru smýšlíme jako o jednoduchém interperu instrukcí - to ale dávno neplatí:




- 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, ...
- typování segmentů – ochrana proti spuštění dat

## *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

## Randomizace uložení do paměti (ASLR)

Různé části adresního prostory (heap, stack, knihovny, …) jsou v paměti umísťovány náhodně
brání zejména proti různým útokům využívajícím buffer overflow

## Další obvyklé prostředky

- Antiviry – dnes majoritně heuristické algoritmy, vzhledem k rychlosti šíření nových útoků nelze spoléhat na bezpečnost
- Intrusion detection / prevention system
- (personal) Firewall – viz kapitola o sítích
- File Insight – shromažďuje historii daného programu v komunitě svých uživatelů
- Code signing – nezaměňovat s kvalitou kódu, pouze do jisté míry zajišťuje identitu autora
- Audit Logy
- Zálohování FS
- Žurnálove FS
- Šifrování FS
- Sandboxing

## *Bezpečný kryptoprocesor*

oddělená implementace kritických funkcí uvnitř samostatné chránění TCB:

- Secure Enclave
- TPM
- Smartcard
- Zvláštní samostatná zařízení (např. LUNA)

Zajišťuje

- hardwarový generátor náhodných sekvencí
- bezpečné genrování klíčů
- bezpečné uložení klíčů
- šifrování a digitální podepisování
- objemové šifrování

na rozdíl od impelentacše v SW kryptoprocesor disponuje detekcí průniku a je schopen zajistit ochranu klíčového materiálu I řízené prostředí pro provádění kryptografických operací

## TPM

- atestace – hash známé SW a HW konfigurace
- binding – svázání kriptografických klíčů s konkrétním zařízením
- sealing – dešifrování proběhne pouzepokud je ověřeno, že systém se nachází ve známém stavu

n adruhou stranu, pokud se najde chyba v kryptoprocesoru, bývá

## Obecný kryptografický modul

FIPS 140-2, ISO/IEC15408 (CC)

- generování klíčů
- kryptografické operace
- potenciálně vysoký výkon
- vlastní politika

výrazně vyšší odolnost vůči útokům, vyšší jistota správně funkce, fyzická bezpečnost pro klíče a výpočet, vynucení funkcí jinak nedosažitelných

## Hardwarový bezpečnostní modul (HSM)

zpravidla kryptograciký modul navržený jako samostatné zařízení s definovaným API

## *Bezpečná úložiště (wault), kryptografické služby*

zahrnutí kryptografických knihoven do služeb OS umožňuje

- tvorbu bezpečných úložišť resp. šifrování souborového systému nebo jeho částí
- systematizované využití kryptografických portokolů
- bezpečnou síťovou komunikaci
- práci s podpisy a časovými razítky
- centrální správu klíčového materiálu, prostředků pro vytváření důvěry

## *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

# Několik vybraných konstruktů

## Virtuální adresní prostor

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 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 běží
poskytovaná ochrana je tedy daleko silnější
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:

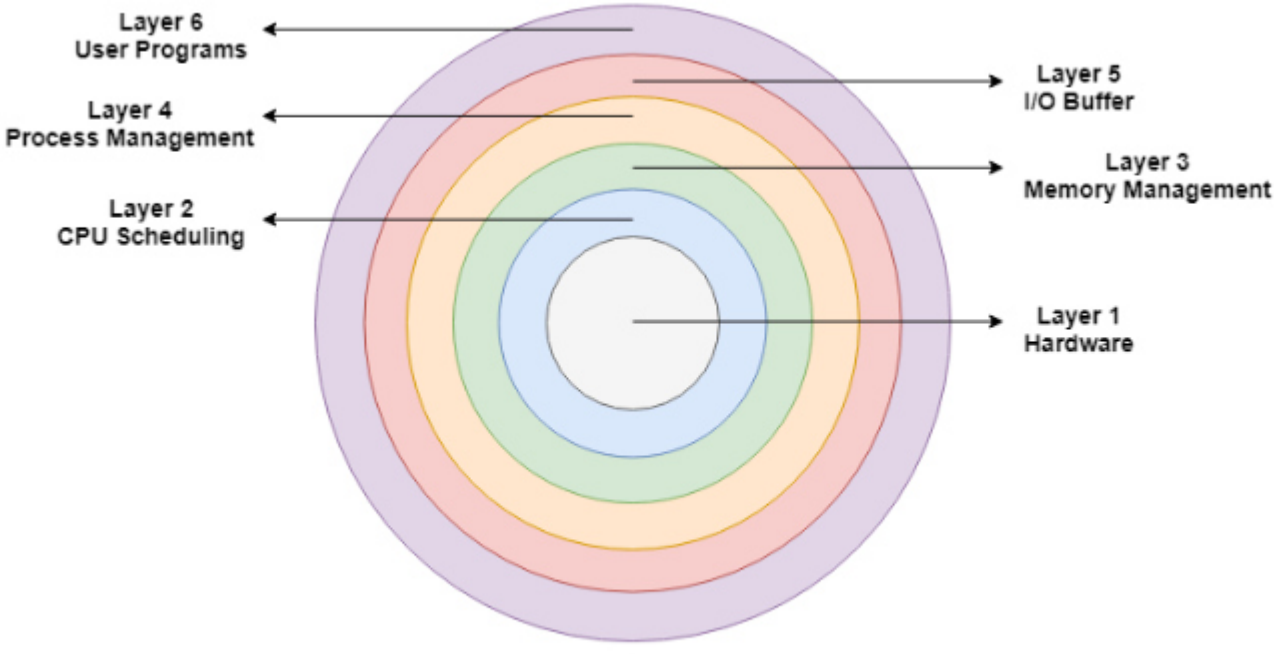
- oddělení od zbytku systému zjednodušuje ochranu mechanismu
- všechny bezpečnostní funkce jsou shromážděny v jednom kusu kódu, tedy implementace bezpečnosti je kompaktní
- kernel nebývá velký, tedy implementace je snadno ověřitelná
- je snazší provádět testování a změny bepečnostního mechanismu
- přes 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




LAYERED OPERATING SYSTEM

## 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.
   - I/O operace obvykle využívají vyrovnávací paměti (buffery) s hrozbou přetečení, či vyčerpání kapacity při nesprávném zpracování vstupních dat

. H

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. obrovským problémem je všudypřítomá honba za výkonem – kešování, předběžné zpracování, pipelining, paralelismus – z toho vyplývající komplikace se synchronizací všech kopií a kontrolou reziduí
6. 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
7. klasickým místem průniku se stávají programy, běžící v rámci OS s rozsáhlými oprávněními
8. vzhledem k současné "prosíťovanosti" světa je častým místem útoku právě obsluha komunikačního protokolu
9. zásadním problémem je pak snaha o zvýšení výkonu při určitých operacích, která vede k obcházení bezpečnostního mechanism, nebo k porušování hierarchického přístupu ke konstrukci OS

## *Hodnocení a eliminace slabin*

v principu dvě strategie –

- kontroly
  - manuální (kontrola návrhu, sledování publikovaných slabin, …)
  - automatické (sw kontroly konfigurace, analyzátory, pentesty, …)
- testování
  - whitebox – testovatel má detailní informace o předmětu testování
  - blackbox – k dispozici pouze obecné informace
  - graybox – částečné informace o předmětu

## *Systémy založené na klientu*

chyby vlastní aplikace na klientu
chyby systému a ostatního SW běžícího na klientu

- Chyby nesprávného používání

    - ukládání dočasných dat na klientu bez zabezpečení
    - používání ne-bezpečných verzí SW (např. se známými chybami, bez podpory, …)
- Chyby komunikace se serverem
    - nesprávná validace identity serveru
    - neprovedení sanitizace přijatých dat
    - chybějící detekce napadení
    - absence validace příkazů a kódu ze serveru

**Postupy**

- Vyhledávání neaktualizovaných komponent OS a aplikací
- Používání zaběhnutých bezpečnostních protokolů v souladu s doporučeními
- Správné techniky programování a použití ověřených knihoven
- Software výhradně se zaručeným původem – code signing
- Progral sledování slabin
- Dodržování postupu pro zabezpečení (hardening)

## *Serverové systémy*

obecně platí totéž co pro klientskou stranu
server nemusí být jednoduché, nebo dokonce možné patchovat

- validovat identitu klientu a identifikovat uživatele přistupujícího prostřednictvím klientu
- klient nutno vždy považovat za nedůvěryhodný (tzn. je nuto sanitizovat vstupy, kontrolovat příkazy, sledovat následnost zpráv, …)
- ochrana serveru před DoS útoky (omezení přenosové rychlosti, množství požadavků, CAPTCHA, podepisování požadavků, …)
- dodržování pravidel bezpečného vývoje (viz dále)
- omezení oprávnění na úložiště na minimum
- logování a analýza logů
- fyzické zabezpečení
- zabezpečení komunikační infrastruktury
- hardening:
    - instalace aktualizací a patchů
    - odstranění nebo uzamčení standardních účtů
    - změna výchozích hesel
    - zákaz nepotřebných služeb, démonů, protokolů a aplikací
    - povolení logů a auditů
    - implementace jen jedné primární služby na server

- úprava výchozích nastavení (povolení FDE, secure boot, …)
- odstranění zdrojových kódů, nepotřebných knihoven, programů, ovladačů, filesystémů, …

## *Databázové systémy*

úspěšně provozovat databáze lze jen nad správně nastaveným OS v prostředí se zajištěnou fyzickou a enviromentální bezpečností

- aplikujte postupy pro hardening
- povolte pouze ty komponenty, které potřebujete
- umístěte datové oblastí a logy na nesystémových partitions
- nastavte oprávnění souborového systému na příslušné adresáře, úložiště, logy, certifikáty, …
- zakažte historii příkazů
- nepoužívejte proměnné environmentu, příkazové řádky či konfiguračních souborů pro zadání hesel/kredenciálů
- nerecyklujte databázové účty
- zakažte anonymní přístup
- všechna spojení pod TLS
- unikátní cetifikát pro každou instanci
- používejte omezená DB view
- odstraňte testovací a vzorové databáze
- změňte všechna výchozí hesla
- dodržujte princip nejmenších oprávnění
- vyhněte se vestavěným rolím
- odstraňte nepotřebné účty
- spravujte všechny účty v souladu s doporučením (best practices)
- povolte logování senzitivních informací, zpracovávejte logy
- přidělte unikátní administrační účty administrátorům (risk-based, role-based přístup, SoD, …)
- používejte bind proměnné pro předávání parametrů
- používejte specifické bezpečnostní mechanismy databáze

šifrování dat v klidu:

- full disk encryption (FDE) – tedy na úrovni média, chrání proti ztrátě zařízení
- šifrování na úrovni souborového systému – obdobné vlastnosti

- transparent data encryption (TDE) – tj. šifrování datových oblastí (souborů, raw devices, …) vlastní databáze, chrání proti útočníkovi s přístupem na souborový systém
- šifrování na úrovni buňky (CLE) – tj. aplikační šiforvání jednotlivých buněk databáze nebo vybraných atributů, umí chránit i proti správcům databáze (se správnou SoD).
- šifrování na úrovni aplikace – tj. aplikace ukládá do databáze již zašifrovaná data

bezpečnostní problémy, se kterými se potýkají databázové systémy

- fyzická integrita databáze - odolnost proti výpadkům napájení, zotavení z poškození
- logická integrita databáze - musí být zachována struktura dat a vzájemné vazby
- elementární integrita - data obsažená v jednotlivých položkách jsou korektní
- 1auditabilita - kdo a jak přistupoval k položkám v databázi
- kontrola přístupu - kdo co může s čím dělat
- autentizace uživatelů - každý, komu je povolen přístup musí být pozitivně identifikován
- dostupnost - uživatelé mohou přistupovat k datům tak, jak jsou oprávněni

## Integrita báze dat

správce báze musí zajistit, že změny dat mohou provádět pouze oprávnění uživatelé
systém musí obsahovat prostředky překonávající nedostupnost položky nebo dokonce celé báze, z hlediska OS a správce systému jde o ochranu relevantních souborů a programů, zálohy, kontroly zařízení atp.
z pohledu SŘBD přistupuje systém transakcí a logů, umožňující rekonstruovat stav databáze

## Elementární integrita

autorizovaní uživatelé mohou vkládat data, ale činí chyby - ty musí SŘBD zachycovat a vyžádat si opravu
metody:

- field checks - test vhodnosti vkládaných dat: zda je to číslo, zda jde o jméno, …
- kontrola přístupu - mechanismus řešící kdo co může měnit, jak naložit s kolizními případy, následnost úprav

- log změn - záznam o všech provedených změnách, obsahuje původní a novou hodnotu
- kontrola čtyř očí
- vícenásobné pořízení dat

## Auditabilita

je třeba vést záznamy o tom, kdo co dělal s kterými položkami - nejen pro to, abychom byli schopni sledovat přístupy a změny, ale i pro dlouhodobé sledování uživatelů a následné rozhodování, zda vyhovět žádosti
je nutné zvolit vhodnou granularitu - bloky, záznamy, položky
zde přistupuje tzv. *pass through problem* - uživatel smí přistupovat k objektu ale tento mu nesmí být předán (např. při vyhledávání)
uživatel může zjistit hodnotu položky i bez přímého dotazu - nestačí log žádostí o přístup k odhadu toho, co ví

## Autentizace uživatelů

SŘBD potřebuje přesně vědět, komu odpovídá
protože však zpravidla běží jako uživatelský proces, nemá spolehlivé spojení s jádrem OS a tedy musí provádět vlastní autentizaci

## Dostupnost

SŘBD má vlastnosti systému a aplikačního programu zároveň - běží jako program a používá služeb systému, pro mnoho uživatelů je však jediným pracovním prostředím
musí rozhodovat současné žádosti více uživatelů
musí být schopen odepřít poskytnutí i nechráněných dat aby nedošlo ke kompromitaci utajovaných informací

## Dvoufázový update

problémem je selhání výpočetního systému během provádění modifikace dat
proces modifikace rozdělíme na dvě fáze:

1. v průběhu první fáze - *záměr* (intent) se provede načtení relevantních dat, uzamčení záznamů, vytvoření pomocných záznamů a kalkulace výsledků
   poslední událostí první fáze je operace *commit*
2. v rámci druhé fáze jsou prováděny trvalé změny dat s ohledem na data získaná v první fázi (= zapsání výsledků)

Pokud některá z fází nedoběhne, může být snadno opakována, po ukončení fáze je systém konzistentní.

# Třífázový update

řešení zápisů a čtení z distribované DB
oproti dvoufázovému update definuje ještě další „nultou fázi“ modifikace
0. ustanovení kvóra – zjišťuje se, je-li k dispozici dostatečný počet instancí distribuované databáze pro provedení operace

kvórum pro čtení – počet instancí, které musí souhlasit s provedením čtecí operace
kvórum pro zápis – počet instancí databáze, které musí souhlasit s provedením zápisu
součet obou kvór musí být větší než celkový počet instancí , kvórum pro zápis > 1/2

Cílem je zabránit stavu, kdy by část distribuované databáze poskytovala zastaralá data, resp. došlo k „rozdvojení“ vývoje datového obsahu

# Zotavení

SŘBD musí vést log o všech akcích, zejména o změnách
v případě havárie potom na základě těchto záznamů a vhodné záložní kopie znovu vygeneruje aktuální stav

# Redundance / Vnitřní konzistence

databáze často obsahují různé redundantní informace za účelem odhalení chyb, zvýšení spolehlivosti, nebo docílení potřebné rychlosti

## Detekční a samoopravné kódy

jde o vhodné způsoby zakódování informací přidáním vhodné redundance tak, aby bylo s co možná největší pravděpodobností detekovat náhodné změny
samoopravné kódy mají navíc schopnost lokalizovat a vyčíslit jisté množství chyb, takže mohou být opraveny
vždy při ukládání je záznam zakódován, při načítání je kontrolována jeho správnost
k dispozici je celá ředa kódů pro různé účely

## Stínové záznamy (Shadow fields)

vybrané atributy nebo  věty jsou uloženy v několika kopiích
v případě nedostupnosti nebo chyby v originálu je použita kopie
metoda účinná leč náročná na prostor

## Paralelismus / Konzistence

SŘBD musí zajistit současný přístup více uživatelů, vyřešit konflikty plynoucí z situací, kdy dochází k současnému zápisu více hodnot do stejné položky, nebo zápisu závisejícím po předchozím čtení položky

## Monitory

jednotky SŘBD zajišťující strukturální integritu databáze - testují, zda vkládaná data typově odpovídají, zda jsou konzistentní s ostatnímy daty v systému atd.

### Porovnání mezí

vkládaná hodnota je testována na příslušnost do jistého intervalu, meze intervalu mohou být určeny i dosti komplikovanou funkcí závisící na jiných hodnotách v databázi
tento test může být rovněž použit pokud je podezření, že uložená data jsou poškozena

### Stavová omezení

popisují podmínky, které musí celá databáze splňovat
nejsou-li stavová omezení splněna, jsou data v databázi vadná
např. ve skladu nelze mít -5,1 rohlíku

### Tranzitivní omezení

jsou omezení, které musí splňovat obsah databáze před provedením určité operace
např. před prodejem 1 rohlíku nesmí být sklad rohlíků prázdný

## SQL injection

Klasický příklad:

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

…a věci se začnou dít, když do proměnné `UserId` zadáte “Beneš OR 2 = 2”
ale lze také namísto OR napsat středník a pokracovat dalším příkazem apod.
Základem řešení je kontrola vstupů a používání SQL parametrů (bind var.) namísto sestavování příkazu v programu

## Problém odvoditelnosti

jde o možnost odvodit senzitivní informace ze znalosti (velkého množství) ne-senzitivních

- Přímý útok - útočník se snaží získat informace přímými dotazy, jejichž odpověď závisí na velmi malém počtu vět, které vyhověly podmínkám dotazu

- Nepřímý útok - často je z databáze obsahující např. senzitivní osobní údaje povoleno zveřejňovat statistické údaje, které není třeba považovat za tajné, z těchto údajů lze ovšem za vhodných okolností získat lP3.čSJSS3WIhFP3.čS

**Náhodný výběr (random sample)**
systém neodpovídá na základě všech hodnot v databázi ale pouze na základě provedeného náhodného výběru ze všech relevantních položek
aby nebylo možné počítáním průměrných hodnot výsledků opakovaných dotazů získat přesné hodnoty, měl by se pro ekvivalentní dotazy používat stále stejný výběr
**Náhodné zmatení (random data perturbation)**
ke každé položce v databázi přičteme náhodnou chybu *e,* přičemž pro opakované dotazy systém zajišťuje, že použitá chyba je stále stejná
pokud je chyba z okolí nuly, je vliv této úpravy na statistické výsledky typu součet, průměr, … malý
metoda je vhodnější než předchozí, neboť lze snáze zajistit stejné výsledky ekvivalentních dotazů.

## *Bezpečnost v aplikačních serverech*

aplikační server využívá OS a databázi jako persistentní repository vlastních dat včetně nastavení bezpečnostního mechanismu
nezbytný vlastní bezpečnostní mechanismus, zahrnující:

- autentizaci
- autorizaci
- auditní záznamy
- správu prostředků
- zajištění dostupnosti
- ochranu komunikace
- konzistenci dat
- řízení změn
- testování
- ...,

Standardem oddělení vývoje od testování (školení, sandbox, ...) a produktivního prostředí
Nezbytností řešit globálně personální politiku a separaci rolí
Vhodné dedikovat kapacity OS(DB) výhradně pro alikační server

## *Technologické řídící systémy*

převážně problém integrity a dostupnosti
zhusta provozováno lidmi bez IT znalostí, často bez porozumění, že zařízení obsahuje IT komponentu

teoreticky by měly být provozovány stejně, jako zbytek IT
specifické problémy:

- obtížnost až nemožnost provést aktualizace firmware (trvalý provoz, zařízení ve vzdálené lokalitě, nemožnost otestovat, zda aktualizace něco nerozbije)
- neprovedení změn výchozích nastavení, hesel apod.
- velmi dlouhá doba životnosti ve srovnání s obvyklým IT vybavením
- závislost na „air-gap“ ochraně sítě

kompenzační opatření

- pokud možno off-line provoz
- pokud možno fyzická ochrana zařízení
- v případě nutnosti připojení je vhodné předřadit proxy, nebo VPN terminátor
- počítače používané pro správu nepoužívat k čemukoliv jinému
- striktní kontrola komunikace zařízení

## *Cloudová řešení*

základem je pochopit rozdělení odpovědnosti mezi poskytovatele a zákazníka v různých modelech

sdílená odpovědnost
- detekce slabin, aplikace aktualizací a patchů
- správa konfigurací
- školení a povědomí

porozumění pro rozdělení odpovědnosti je klíčové pro udržení bezpečnosti
pro posouzení bezpečnosti třeba zvážit povahu cloudu – veřejný / privátní / komunitní / hybridní: ne nutně je provátní či on-prem cloud bezpečnější, spíše se trochu liší model ohrožení

## *Distribuované systémy*

… rozumíme geograficky rozložený soubor prostředků / systémů fungující navenek jako jediný systém
podstatně větší expozice pro útok
mimořádná důležitost zajištění komunikace (viz kapitola o sítích)
specifické bezpečnostní problémy:
- autentizace a zajištění utajení všech spojení mezi komponentami
- ochrana proti DoS útokům
- riziko nedostatečné homogenity
- zachování konzistence v případě výpadku spojení (viz např. třífázový update)
- zajištění shodných bezpečnostních opatření všech částí geograficky distribuovaného prostředí
- rozdílné požadavky na soukromí a datovou suverenitu v různých jurisdikcích

## *Internet věcí*

velmi podobné problémy technologickým řídícím systémům
„spravováno“ a provozováno začasté lidmi bez jakéhokoliv technického, natož bezpečnostního povědomí
vývoj:
- měl by zahrnovat modelování rizik a hrozeb
- kontrolu bezpečnostní architektury
- dodržovat pravidla pro vývoj bezpečného SW (viz dále)
- vytvořit podmínky pro bezpečné aktualizace – nejlépe automatické ve výchozím nastavení
- zajistit výměny výchozích kredenciálů (např. nepřipojením se k internetu s výchozími hodnotami)

opět je důležité oddělit IoT zařízení od ostatní infrastruktury a kritických systémů
použití:
- nutno změnit výchozí kredenciály před zahájením používání

- udržovat zařízení aktuální
- neumísťovat IoT zařízení do prostředí otevřeného do Internetu
- oddělit segment sítě s IoT zařízeními od ostatních systémů
-

- zabezpečení hostitelského operačního systému
- minimalizace běžících služeb, zejména webserverů, mailerů apod.
- omezení komunikace mezi kontejnery na nezbytné minimum

## *Bezserverová řešení*

… služby jako Azure Functions, Google Cloud Functions apod.
poskytovatel provozuje a řidí servery a poskytuje de facto aplikační služby na požádání
uživatel služeb je zodpovědný za vývoj a bezpečnost příslušného SW, zbylé věci řeší poskytovatel
výhodou bývá lepší řízení přístupu k jednotlivým službám
rizika a opatření

- správné nakládání s autentizačními informacemi
- správné nastavení oprávnění
- zajištění integrity prováděného kódu
- správný monitoring
- detekce podezřelých událostí
- monitoring provozu

## *Embedded systémy*

překrývá se s IoT
obecně se rozumí prostředky zpracování informací připojené k většímu zpravidla mechanickému zařízení (ATM terminál, automobil, medicinská zeřízení, automatizace budov, …)

útoky lze rozdělit na útoky proti

- uživatelskému rozhraní – hledání kombinací tlačítek, zacházení v rozporu s návodem, hledání vstupu do administračního módu, nesmyslné vstupy…
- fyzické útoky – porušení krytu či obalu
- útoky na seznory
- manipulace s výstupem - přímá manipulace s ovládaným zařízením
- útoky na zpracování dat

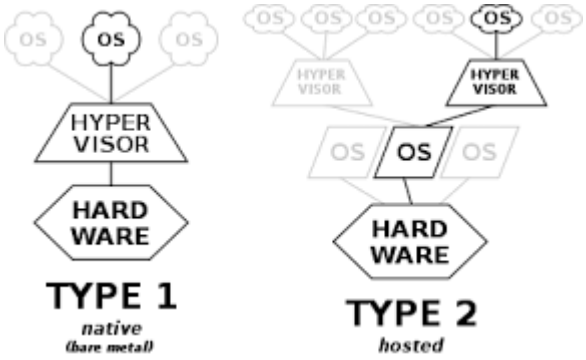
opatření

- defenzivní návrh předpokládající útok
- pentesty
- lepší krytování

- ochrana vnitřních spojení
- detekce otevření
- ochrana proti výměně firmware

## *Virtualizované systémy*

i zde třeba rozumět rozdělení odpovědnosti za bezpečnost
poskytovatel zpravidla spravuje virtualizační platformu a přístup k jejímu ovládání




obecně menší expozici představuje hypervisor typu 1
rizikem navíc je útok na hypervizor, nebo nebezpeči chyby při administraci – nepsrávná konfigurace přidělení prostředků, (lokální) nedostatek zdrojů, technická chyba hypervizoru, obecně složitější prostředí
výhody plynoucí z možnosti relokovat aplikaci, dynamicky přidělovat zdroje, lepší testování a prototypování, snazší zálohy, …
opět nuné:

- správná konfigurace hypervisoru
- správný monitoring
- detekce podezřelých událostí
- monitoring provozu

# Bezpečnost prostředí a zařízení

zabránění neautorizovanému vstupu a výstupu
dostupnost proti environmentálním a infrastrukturním problémům

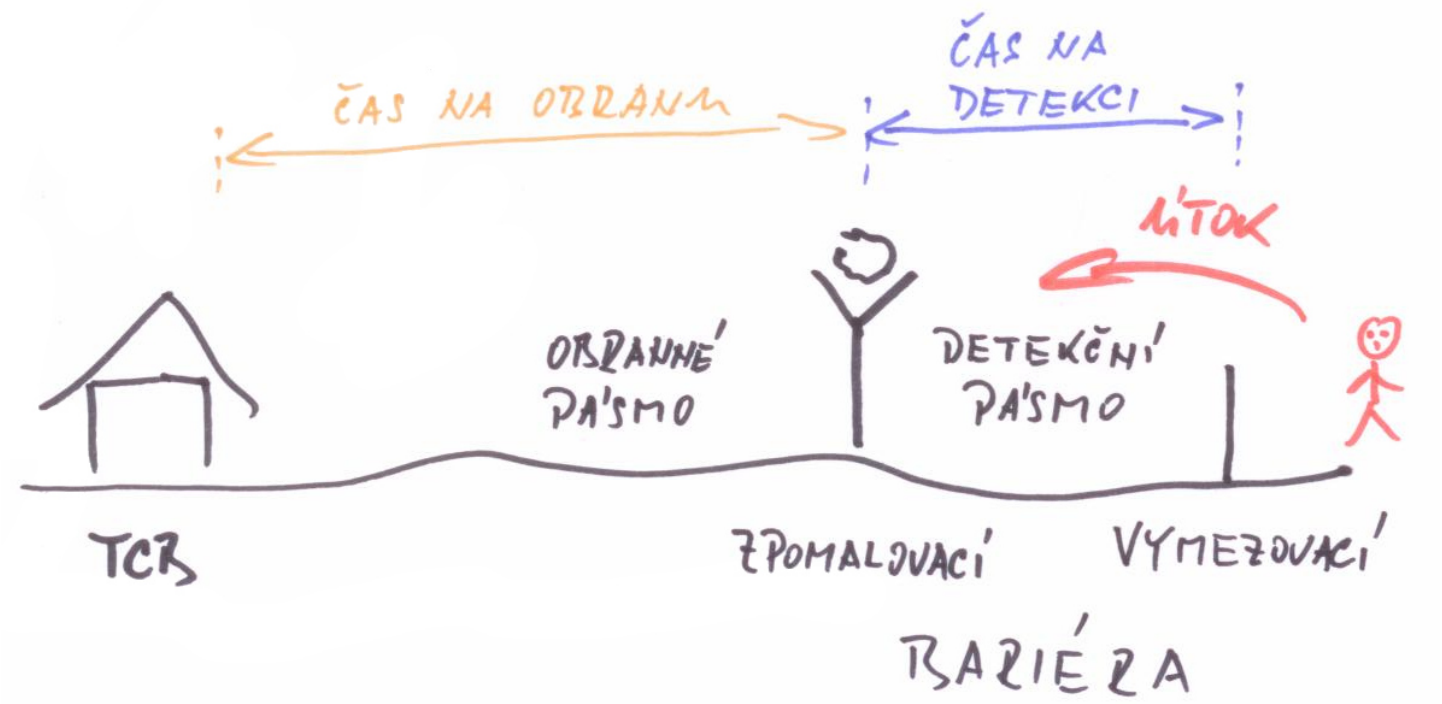
fyzická ochrana se snaží eliminovat případnou hrozbu ještě dříve, než přijde do přímého kontaktu s vlastním výpočetním systémem
primární nebezpečí která fyz. ochrana pokrývá můžeme rozdělit:

- přírodní katastrofy
- vandalové

- nehody
- ostatní vnější vlivy
- cílevědomí pachatelé

a dále vytváří předpoklady pro aplikaci dalších protiopatření

… smyslem je vytvořit uzavřené prostředí (TCB), ve které jsme schopni zajistit požadované vlastnosti prostředí odpovídající bezpečnostní politice




Uvedené schéma platí obecně, ať je TCB fyzicky realizována jako rozsáhlý komplex budov (chráněná základna), jediná místnost (serverovna, režimové pracoviště) či jediný čip (třeba čip v e-občance, nebo sim karta).

## *Přírodní katastrofy*

není jim možné předcházet, je třeba se soustředit na omezení možného odpadu a na odstranění případných následků
základem pasivní obrana – tj. pokud možno umístit IS mimo zónu pravděpodobného vlivu

### Záplavy

v podstatě dvou druhů
stoupající voda - většinou bývá dost času odstavit systém a přinejmenším datové nosiče přesunout do bezpečí. Pro tyto účely by každá komponenta systému měla být jasně vyznačen stupeň důležitosti, aby s odsunem mohli efektivně pomáhat i nekvalifikovaní pracovníci.
padající voda - např. poruchy potrubí, izolací apod. Tyto záplavy bývají velmi rychlé, v první fázi stačí vhodný nepromokavý kryt a následný odsun zařízení. Opět vhodné vyznačení stupňů důležitosti komponent.

## Požáry

představuje často nebezpečí nejen pro techniku, ale i pro obsluhující personál
je vhodné mít vyzkoušen postup zahrnující bezodkladné odstavení systému a evakuaci personálu a životně důležitých komponent systému
je třeba vhodně volit automatické protipožární systémy chránící prostory výpočetního systému, vhodné je umístit nejdůležitější části systému v prostorách s vysokou pasivní požární bezpečností

## Poruchy napájení

pro nejdůležitější části systému je nutné zajistit náhradní zdroje energie, které jsou v případě výpadku hlavního napájení schopny dostatečně rychle zajistit dodávky elektřiny
na kratší dobu jsou to různé akumulátory a UPS zdroje, v případě potřeby překonávat delší výpadky pak agregáty na výrobu el. energie
důležité jsou rovněž filtry a přepěťové ochrany chránící zařízení před výkyvy napětí, blesky apod.

## Chlazení

některé komponenty jsou citlivé na teplo, při ztrátě chlazení může dojít k jejich zničení
větší systémy nelze bez odpovídajícího chlazení provozovat ani krátkodobě, příslušná technologie se tak může stát klíčovou komponentou

## Hmotnost

serverová technologie může vyžadovat speciální podlahy se zvýšenou nosností, může se stát problémem zejména při zařizování záložního centra po havárii
jindy lze hmotnost využít jako bezpečnostního mechanismu

## Prašnost, vibrace, další vlivy

... mohou i výrazným způsobem omezovat životnost komponent, případně zvyšovat poruchovost

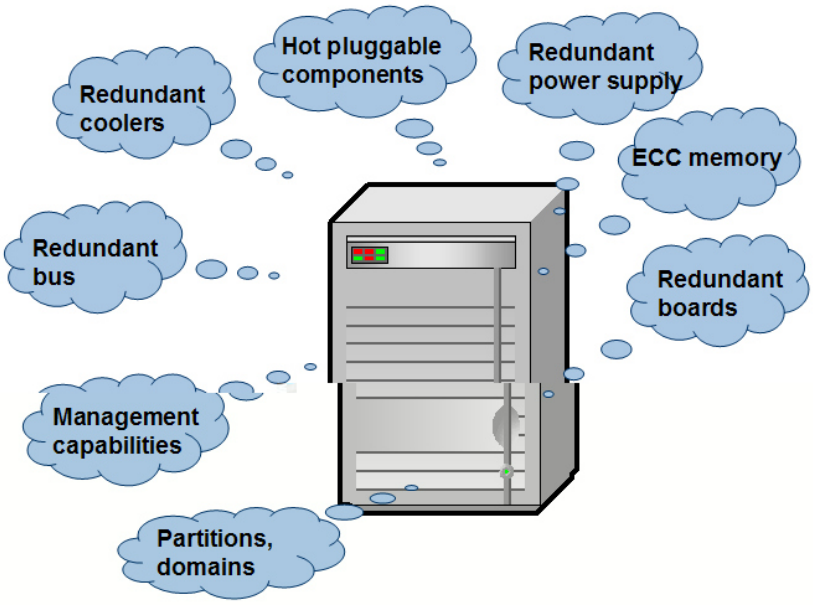
# *Enviromentální bezpečnost*

… tzn. zajištění fyzikálních podmínek slučitelných s provozem systému

napájení
- redundantní připojení
- UPS – nepřerušitelné zdroje
- generátory
- ochrana proti přepětí
- ochrana proti podpětí
- ochrana proti VF rušení

| | |
|---|---|
| | - zajištění stále frekvence |
| klimatizace | - chlazení |
| | - zajištění proti kolísání teploty |
| | - zajištění stále vlhkosti |
| | - udržování nízké prašnosti |
| | - eliminace agresivních látek |
| | - udržování stálého (pře)tlaku |
| protipožární ochrana | - ochrana osob |
| | - automatické hasicí systémy |
| | - systémy protipožárního odvětrávání |
| | - systémy havarijního odstavení |
| | - pasivní protipožární bezpečnost |
| | - ochrana před sáláním |
| ochrana proti záplavám – statická bezpečnost | |
| | - (automatické) uzávěry |
| | - protipovodňová opatření |
| ochrana proti vibracím | |
| protiradiační ochrana | |

## *Funkčnost vlastního HW*




## *Datová persistence*

Níže uvedené mechanismy se doplňují, spíše než nahrazují

# Zálohování

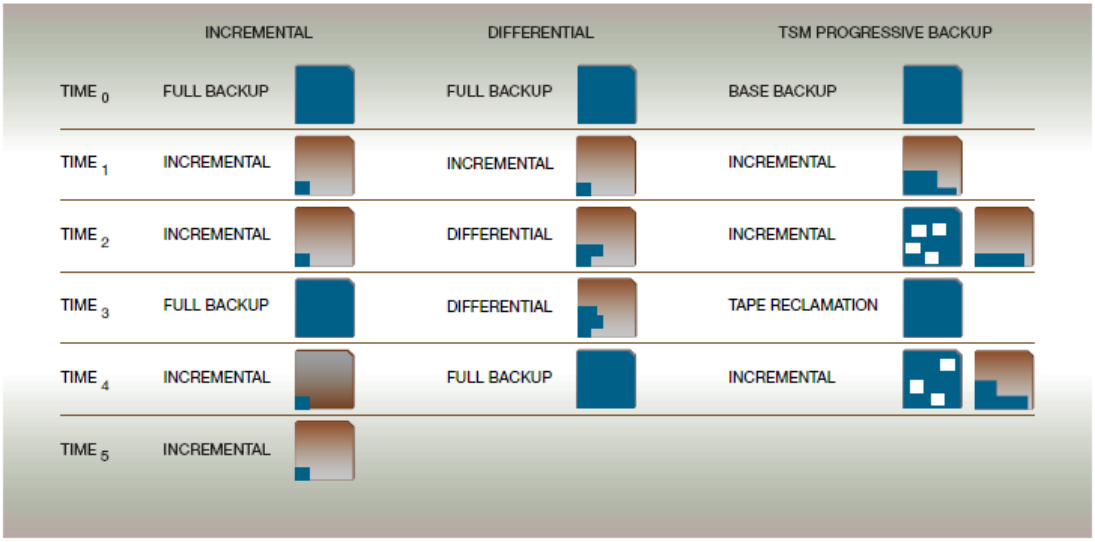
technologie flash copy, instant flash, shadow copy apod.
nezbytností management zajišťující plánování záloh, podporu restore, evidenci záloh

- off-line konzistentní zálohy
- on-line zálohy

- on-site zálohy
- off-site zálohy (archivy)

Příklady zálohovacích strategií:

- Inkrementální zálohy
- Diferenční zálohy
- Deduplikace
- Komplexní systém řízení záloh (zde Tivoli Storage Manager)




# Disková pole typu RAID

| | |
|---|---|
| Level 0 | Striping – data jsou rozložena po částech na všech discích, neposkytuje ochranu proti HW závadě. |
| Level 1 | Mirroring – veškerá data replikována na separátních discích. |
| Level 2 | Ochrana pomocí Hammingova kódu – na dalších discích se ukládá redundance umožňující opravu 1-bitových chyb a detekci 2-bitových. Prokládání po bitech či blocích. |
| Level 3 | Datové bloky jsou rozkládány na disky pole, na vyhrazený disk se zapisují paritní bity. |
| Level 5 | Datové bloky rozloženy podobně jako při stripingu, ale paritní bloky jsou distribuovány přes všechny disky. |

Pole obvykle obsahuje tzv. hot-spare disky, které nejsou aktuálně používány, ale využijí se jako operativní záloha v případě poruchy některého z disků, na která se ukládají data.
Důležité další vlastnosti pole – **flash copy, instant copy, remote replication**

## *Zachování funkčnosti aplikace*

Co se dá dosáhout

- extrémně rychlý návrat
- omezení rozsahu výpadku
- transpatrentnost výpadku

typy clusterů

- hot – standby
- active - active

Základní strategie

- Fault tolerance
- Load ballancing
- Virtualizace

## *Prostorová ochrana*

prostředky umožňující zabránit potenciálním útočníkům ve vstupu do prostor, kde jsou instalovány důležité komponenty systému, nebo vnesení potenciálně nebezpečných předmětů či detekci takové skutečnosti, případně může být cílem zabránění opuštění prostoru, respektivě vynesení komponent
předmětem útoku můžou být:

- vlastní počítače
- výměnná záznamová media
- části počítačů (myši, světelná pera, …)
- vytištěné senzitivní materiály
- části síťové technologie,
- klíčoví pracovníci

metody ochrany:

### Stráže

měly by být k dispozici nepřetržitě, musí osobně znát všechny pracovníky, nebo musí umět rozpoznat oprávněnou osobu jiným způsobem - např. dle tokenu
stráž musí provádět záznam o pohybu všech osob

problémem jsou zaměstnanci, se kterými byl nedávno rozvázán pracovní poměr a celková míra spolehlivosti stráží

náhradou nebo doplňkem stráží mohou být různé turnikety a přechodové komory vybavené zařízeními pro identifikaci osob nebo tokenů

## Elektronická prostorová ochrana

- dveřní a okenní kontakty - detekují otevření
- otřesové hlásiče - detekují rozbití nebo proražení střežené plochy - skla, příčky, přepážky, …
- vodičové desky, drátěné sítě - slouží k detekci průrazů ve stěnách, podlahách apod.
- kontaktní matice - při instalaci pod podlahové krytiny slouží k detekci vstupu osob do chráněného prostoru
- Mikrovlnné, ultrazvukové, infračervené detektory - reagují na změnu rep. přerušení svazku příslušného záření
- zvukové hlásiče - reagují na specifické zvuky jako řezání, vrtání, šroubování, …
- kyvadlové hlásiče - reagují na otřesy a vychýlení z původní roviny

k ochraně jednotlivých předmětů lze použít obrazových vah, závěsů apod.
vhodným prostředkem v mnoha případech je průmyslová televize, zejména v kombinaci se záznamem snímaného obrazu

## Detekce výstupu

vhodnou metodou je pokoušet se odhalit člověka odnášejícího část vybavení
prostředky jsou obdobné, jaké se používají v obchodních domech - různé nálepky či přívěsky, které lze snadno detekovat

# *Klasifikace serveroven*

| *Tier* | *Dostupnost* | *Redundance* |
|---|---|---|
| 1 | 99,671 | Žádná, více SPOF |
| 2 | 99,741 | Částečná Neredundantní UPS |
| 3 | 99,982 | N+1 Možnost vyjmout libivolné zařízení mimo provoz (opravy, údržba) |
| 4 | 99,995 | 2N, Žádný SPOF, automatická kompenzace jediné závady na libovolném zařízení |

## *Likvidace medií se senzitivními informacemi*

je nutné mít k dispozici prostředky ke zničení nebo znehodnocení medií před jejich exportem z chráněného perimetru, nikdy není jasné, kam se dostanou

### Zkartovače

existují v mnoha verzích pro nasazení v různých stupních zabezpečení, navzájem se liší jemností a způsobem provádění skartování
slouží především k ničení papírových dokumentů, dále disket pásek ze streamerů, kazet, barvicích pásek z úderových tiskáren

### Přepisování magnetických medií

prosté smazání souboru většinou vede pouze k odstranění záznamu o jeho existenci, proto je nutné zajistit skutečné fyzické přepsání původních dat, pro větší stupeň bezpečnosti několikanásobné
metoda je zdlouhavá a ne úplně bezpečná

### Degaussery

přístroj vygenerováním silného elektromagnetického pulsu dokáže zničit původní magnetické pole
ani v tomto případě nejde o zcela spolehlivou metodu vhodnou pro nasazení v nejvyšších stupních utajení

## *Odpovědnost za zabezpečení*

celkovou odpovědnost má vedení organizace, lze ji rozdělit na odpovědnost za návrh bezpečnostní strategie a na odpovědnost za dodržování bezp. opatření
důležitou součástí bezpečnosti jsou opakované namátkové kontroly

# Elektromagnetické vyzařování

… je způsobeno změnou proudu ve vodiči
problémem je vyzařování nejrůznějších částí počítačů, zejména monitorů a přenosových linek, důležité informace mohou unikat i naindukováním do napájecích obvodů zařízení
odposlech elmg. záření začasté není možné kriminalizovat a je nutno se vyrovnat s faktem, že jej útočník provádí
metody ochrany:

- Vzdálenost - intenzita vyzařování klesá se čtvercem vzdálenosti
- Zmatení - množství podobných signálů ztěžuje odposlech, je možno generovat záření podobných vlastností jako vyzařování výpočetního zařízení, nebo umístit více vyzařujících zařízení blízko sebe

- Speciální vybavení - je možné zakoupit součásti vyvinuté tak, aby jejich vyzařování nepřekračovalo jistou únosnou mez
- Vhodné umístění - alternativou nákupu nevyzařujících zařízení je umístění standardních zařízení v prostorách, které zamezují šíření záření - jde o různé schránky, skříně případně celé stíněné místnosti