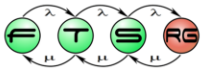
Operációs rendszerek (vimia219)

Hitelesítés és engedélyezés

Tóth Dániel, Micskei Zoltán



Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Utolsó módosítás: 2012. 05. 07.

Számítógépes rendszerek biztonsága

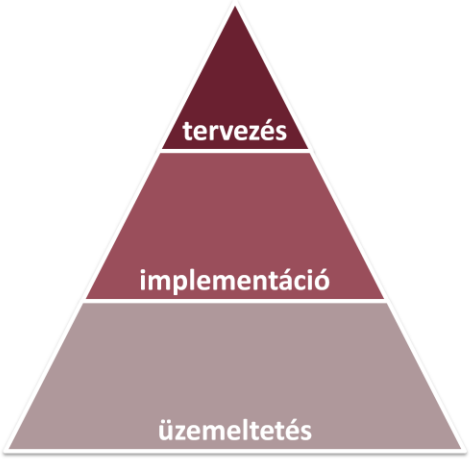
- Fontos ez?
- Mindenkinek fontos?
- Mikor fontos?

Általános kezdés:

- Nyilvánvaló, hogy banki, üzleti szférában fontos a biztonság, de máshol?
- Otthoni gépen? Személyes adatok megszerezhetőek stb. vissza lehet élni vele -> igen tényleg fontos.
- Beágyazott, zárt rendszerekben minek foglalkozni vele?! – itt jön a zombis kép – hát azért, mert a rendszerek sosem tekinthetőek teljesen „zártnak”, úgyis megpróbálnak majd belepiszkálni. Erre veszélyesebb példa a lengyel srác esete, aki a Lodz-i villamossín váltókhoz készített távirányítót és kisiklatott néhány villamost. Biztonságosnak tekinthető-e egy tűzfallal védett belső hálózat? Milyen buktatói lehetnek?
- Egy képszerkesztőben, médialejátszó vagy dokumentum néző alkalmazásban fontos-e a biztonság? Meglepő módon igen, mert ellenkező esetben a felhasználó által elvárttól eltérő viselkedés kényszeríthető ki belőle, egy szándékosan hibásan formázott bementtel.

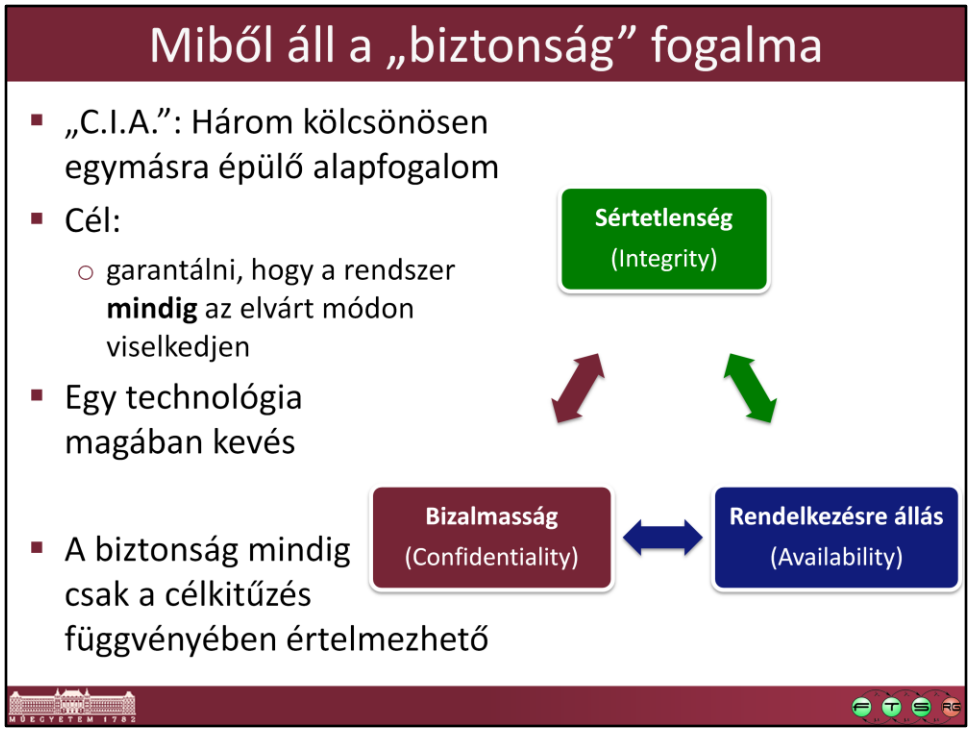
Mikor fontos a számítógépes biztonság?

- Szoftverfejlesztésben minden szinten foglalkozni kell a biztonsággal
- *„Ha egy rendszert nem terveztek biztonságosra, akkor később lehetetlen azzá tenni.”*
- A rendszer biztonsága a leggyengébb láncszem biztonságával azonos. *„Az operációs rendszer nem csodaszer, rosszul megírt alkalmazás ellen nem véd.”*



Későn megjelent biztonsági követelmények csak drágán és korlátozott mértékben implementálhatók. Pontosan úgy, mint minden más szoftver hibánál: minden szinten lejjebb lépve exponenciálisan nő a hiba kijavításához szükséges ráfordítás. Csakhogy itt annyival is rosszabb a helyzet, hogy nehezen értékelhető ki, hogy végül is mennyire lett biztonságos a rendszer a javítás után.

Régen rossz (ámde sajnos roppant tipikus), ha az üzemeltetés szintjén kell megoldani ezt a problémát. Erre vannak mindenféle hardveres trükkök (NX bit), operációs rendszer szintű intézkedések (főleg Openbsdben vannak implementálva, Linuxhoz is van ilyen patch készlet: PAX, grsecurity), futtató wrapperek stb. Ezek inkább csak többé-kevésbe hatékony kényszermegoldások az implementációs hibák ellen, tervezési hiányosságok ellen nem véd.

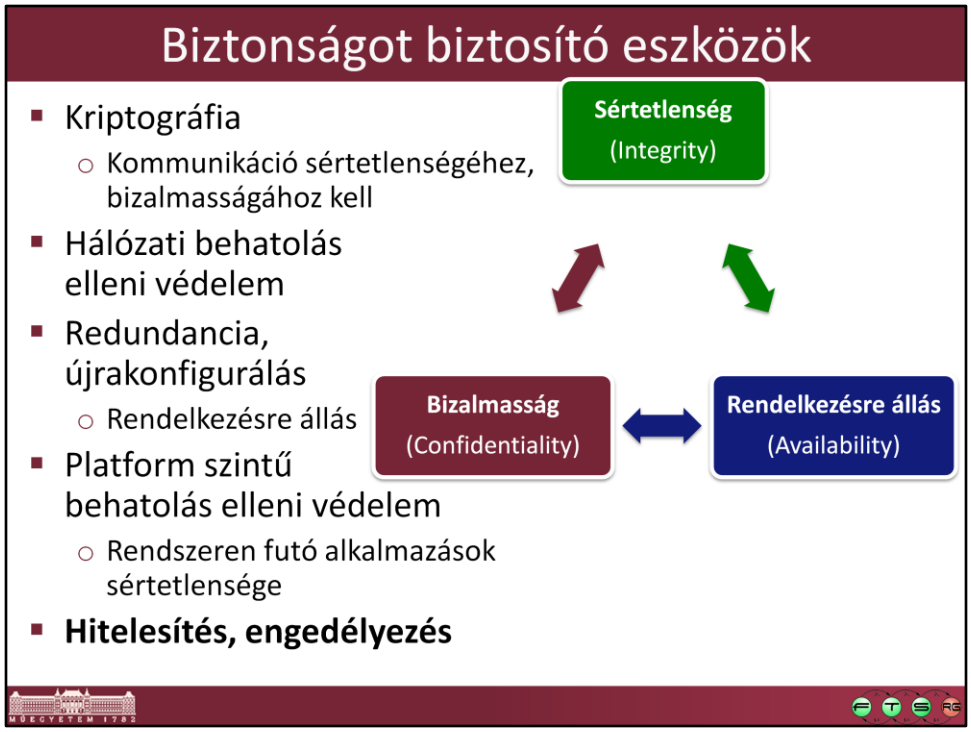


Fontos általános megjegyzés: most a Security világ szemszögéből tekintjük át, a dependability-s világ más elnevezéseket és csoportosítást használ

- *Bizalmasság*: titoktartás harmadik féllel szemben, először mindenkinek ez jut eszébe a számítógépes biztonságról
- *Sértetlenség*: legalább olyan fontos, hogy az üzenetek feladója tényleg az legyen, aki a feladó mezőben szerepel, illetve a kapott üzenet tartalma pontosan az legyen, amit az (igazi) feladó küldeni akart. (Hitelesség, Authenticity). Néha külön kiemelik a letagadhatatlanságot (Non-deniability), vagy kifejezetten a letagadhatóságot. Továbbá a rendszer elemei is megmaradnak olyannak, amilyennek elvárjuk, rongálástól, illetéktelen módosítástól mentesnek. (Tamper resistance).
- *Rendelkezésre állás*: gyakran elfeledett része a biztonságnak, kárt lehet okozni azzal is, ha csak működésképtelenné válik egy rendszer (Denial of Service), különösen fontos „biztonságkritikus” rendszerekben, ahol emberélet függhet a rendelkezésre állástól.

A három tulajdonságot külön-külön kell vizsgálni, nem feltétlen következménye egyik a másiknak (pl. egy digitális aláírással ellátott levél garantálhatja a sértetlenséget, de ha nincs titkosítva, akkor nem garantálja a bizalmasságot).

Megjegyzendő, hogy a hagyományos hibatűrésben a természet okozza a hibákat, ezeknek megfigyelhető, számítható valószínűsége van, hibatűrési mechanizmusokkal (redundancia beépítése) ez a valószínűség tetszőlegesen csökkenthető. A hibatűrési mechanizmusok sértetlenséget tételeznek fel. A szándékos támadások esetén az ellenfél intelligens, minden 0-tól eltérő valószínűségű „hibát” előállíthat.



(a lista nyilván nem teljes, más szempontok szerint is csoportosítható)

- Kriptográfia: Kódolástechnika c. tárgyban kerül elő. Szükséges eszköz a sértetlenség és bizalmasság garantálásához, de önmagában nem elég.
- Hálózati behatolás elleni védelem: cím alapú szűrések, tűzfalak, forgalom megfigyelés, rögzítés, gyanús forgalom beazonosítása – mindhárom biztonsági cél esetén szükség lehet rájuk, támadás hatásait hálózat szintjén próbálja behatárolni
- Redundancia, újrakonfigurálás: véletlen meghibásodás vagy támadás esetén a működőképesség fenntartása vagy helyreállítása
- Platform szintű behatolás elleni védelem: memóriavédelem, futási szintek, NX bit, stack ill. heap randomizáció, rendszerhívás monitorozás és korlátozás – olyan megoldások, amik hibás alkalmazások esetén próbálják behatárolni egy támadás hatásait, OS illetve futtató hardver szintjén
- Hitelesítés, engedélyezés: erről fog szólni a most következő előadás

„Ne lehessen illetéktelenül adatokhoz jutni, műveletet végezni” – mit jelent az illetékesség/illetéktelenség?

- Üzenetalapú kommunikációként modellezünk most minden műveletet, beleértve a gépen belül illetve a felhasználó és gép közötti interakciót is. Feltételezhetjük, hogy az üzenetek lemásolhatóak, átírhatóak anélkül, hogy ez észlelhető lenne, beleértve a az üzenet küldőjének azonosítóját is. Tennünk kell annak érdekében, hogy megbizonyosodjunk arról, hogy az üzenetet valóban az küldi (a műveletet az kezdeményezi, stb.), akinek mondja magát. Ez a feladat a hitelesítés.
- Ha hiteles a küldő, akkor még mindig eldöntendő kérdés, hogy neki szabad-e elvégeznie a műveletet. Ez a feladat az engedélyezés.

Itt is a két részterület együttesen biztosítja a cél elérését.

(Megjegyzés: az authorization szót szokták máshogy is fordítani, pl. meghatalmazás, feljogosítás, jogosultság-ellenőrzés)

Tartalom

- Számítógépes biztonság bevezető
- **Felhasználók kezelése, hitelesítés**
  - UNIX, Linux alatt
  - Windows alatt
- Engedélyezés
  - Engedélyezés általános sémái
    - Szerep alapú hozzáférés-vezérlés
    - Hozzáférési jogosultság listák
  - Engedélyezés UNIX, Linux alatt
  - Engedélyezés Windows alatt
    - Biztonsági alrendszer alapok
    - Központosított hozzáférés-vezérlés

Ezekről majd a következő előadáson részletesen

Hitelesítés

- Mi alapján dönthető el, hogy ki kicsoda?
  - …amit tud (pl.: jelszó)
  - …amije van (pl.: kulcs, belépőkártya)
  - …ami ő (pl.: ujjlenyomat, arckép)
- Ezek (akár egy kombinációja) alapján egy gép el tudja dönteni, hogy ki a személy, aki előtte ül
  - Mi a helyzet a gép-gép közötti szolgáltatásokkal?

Hitelesítés

- Hitelesítés három szinten kerülhet elő:
  - Ember és gép közötti interakció
  - Gépen belül futó alkalmazások valamint az OS között
  - Gép és gép között valamilyen hálózaton át
- Hitelesítési *protokollok* kellenek
  - Gépen belül ill. gépek között csak az „amit tud” séma
  - Használható bonyolult kriptográfiai számítás (embernél fejben nyilván nem)

Operációs rendszer esetén protokollnak tekinthető az API, a folyamatok azonosítása, hozzátartozó engedélyeinek kezelése. Tágabb értelemben véve az OS szolgáltatása lehet, hogy hitelesítéssel garantálja a rá telepített programok sértetlen állapotát (aláírások ellenőrzése), OS szintű szolgáltatás lehet még a hálózati kapcsolatok hitelesítése, bizalmasságának garantálása is (pl. SSL könyvtár). Ez értelmezés kérdése.

Miből áll egy felhasználói fiók?

User

+ ID
+ Name
+ Real Name
+ Personal data…
+ Shared Secret (Password, etc.)
+ Private Datastore path

A rendszer számára a felhasználó egy objektum…

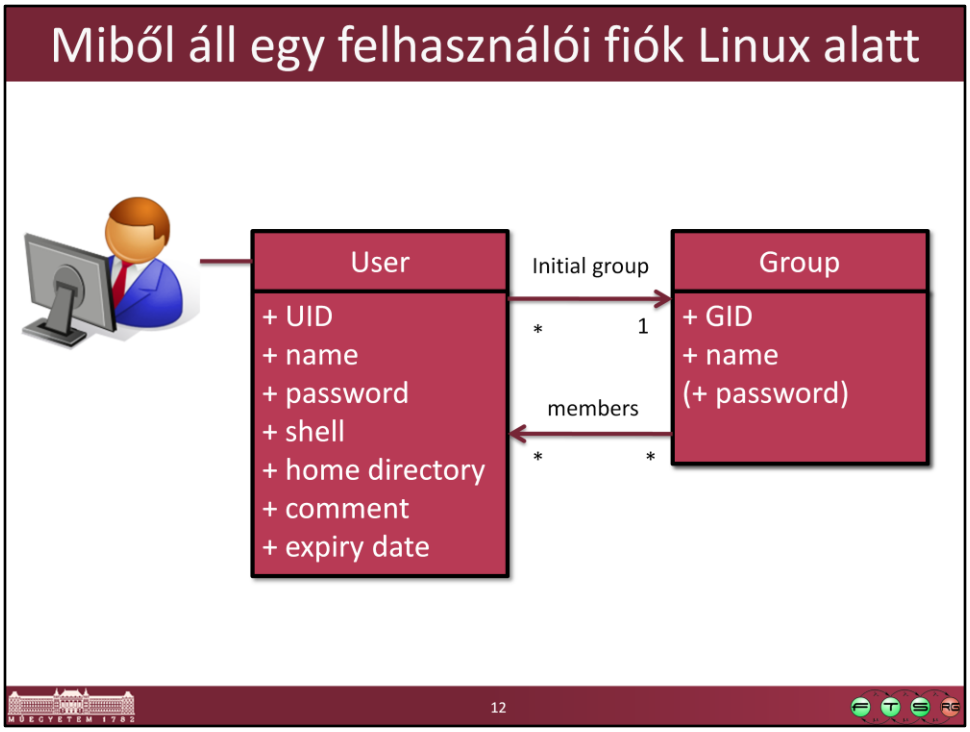
Miből áll egy felhasználói fiók Linux alatt

User

+ UID
+ name
+ password
+ shell
+ home directory
+ comment
+ expiry date

Fiókot azonosítja UNIX/Linux esetén:

- UID (integer)
- (root 0, többiek 1000-…)



Meglehetősen ritkán használt funkció a csoporthoz hozzárendelt jelszó, de van ilyen is.

DEMO Felhasználói fiókok Linux alatt

- Segítség: `man` parancs
  - `man 5 passwd`
- Felhasználó és csoportok tárolása fájlokban:
  - `/etc/passwd`
  - `/etc/shadow`
  - `/etc/group`
- Létrehozás, módosítás, törlés
  - `useradd, usermod, userdel`
  - `groupadd, groupmod, groupdel`
  - `passwd`

1. passwd fájlban vannak a felhasználói adatok, ezt mindenki olvashatja (hogy pl. az UID -> account name leképezést meg tudják csinálni)
[meres@irfserver ~]$ man 5 passwd

- Ez megadja, hogy milyen mezői vannak a /etc/passwd fájlnak

[meres@irfserver ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:2:2:daemon:/sbin:/sbin/nologin
meres:x:500:500:Teszt Felhasznalo:/home/meres:/bin/bash
…

- Látszik a 0-s UID a root felhasználó
- A bin nevű felhasználó a rendszer része, ezzel például nem is lehet belépni (a nologin nevű program az alapértelmezett shellje, ami kilép)
- A meres pedig egy saját felhasználó már

2. shadow fájlban vannak a tikosított jelszavak
nezzuk meg, hogy milyen mezok vannak a shadow fajlban
[meres@irfserver ~]$ man 5 shadow

a shadow faljt mar csak a root tudja megnezni
[root@irfserver ~]# cat /etc/shadow
meres:thWUfjWdVy4MkvhmBO7UJXgEgpCyHQJUaD0:15380:0:99999:7:::

- Ez tárolja a titkosított jelszó mellett a legutolsó jelszómódosítás idejét, a jelszó lejárati idejét stb.

3. a group fájl tárolja a rendszerben lévő csoportokat
[meres@irfserver ~]$ cat /etc/group
root:x:0:root
adm:x:4:root,adm,daemon
meres:x:500:

- A csoportnak is lehet jelszava, van egy azonosítója, és utána következnek a tagjai

Identitás váltása

- Folyamat identitása menet közben változhat
  - Real UID vs. Effective UID
- `id`
  - Ki vagyok én?
- `su`
  - váltás egy adott felhasználóra
  - adott felhasználó jelszavát kéri
  - **`su -`** Login shellt indít (környezeti változók miatt fontos)
- `sudo`
  - parancs végrehajtása más nevében
  - saját jelszót kéri + jog kell hozzá (lásd `/etc/sudoers`)

Részleteket lásd pl. The GNU C Library: Users and Groups (http://www.imodulo.com/gnu/glibc/Users-and-Groups.html)

DEMO Futó folyamatok

- Ki vagyok én?
  - `id`
- Hogyan állapíthatjuk meg egy futó folyamatról, hogy ki a tulajdonosa
  - `ps aux, pstree, /proc/$PID/status`
- Folyamat futása közben effektív user és group váltása
  - `su, sudo` parancsok

- nézzük meg a su és su - közötti különbséget:
  echo $PATH
  más-mást ad vissza

Hitelesítési mechanizmusok Linuxon

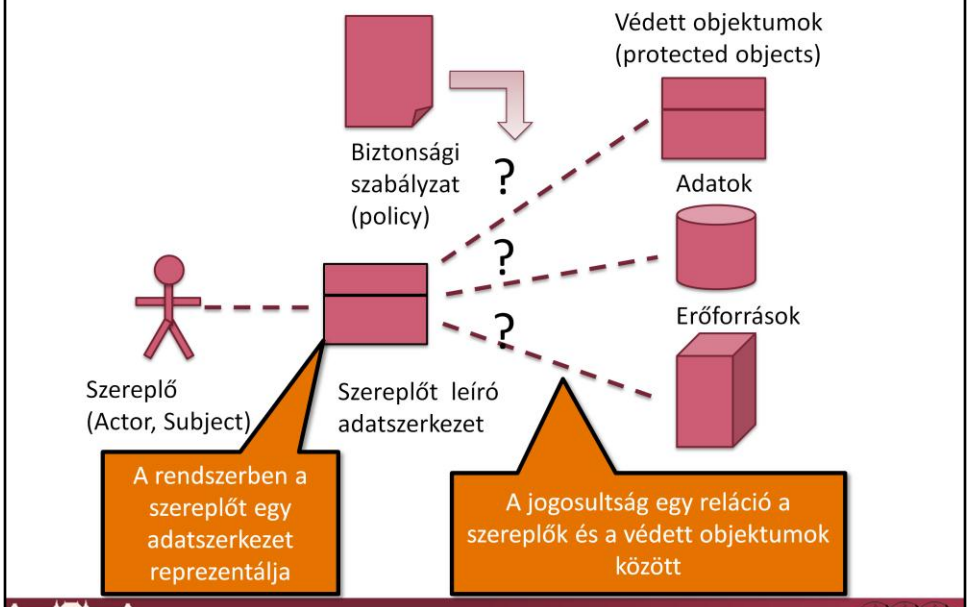
- Pluggable Authentication Modules (PAM)
  - Hitelesítési mechanizmusoknak keretrendszer
  - Cél: alkalmazástól leválasztani a mechanizmusokat
  - Mechanizmusok dinamikus betöltése
  - Mechanizmusok: `/lib/security/`
- Kerberos
  - Hálózati hitelesítési protokoll
  - Lásd RFC 4120

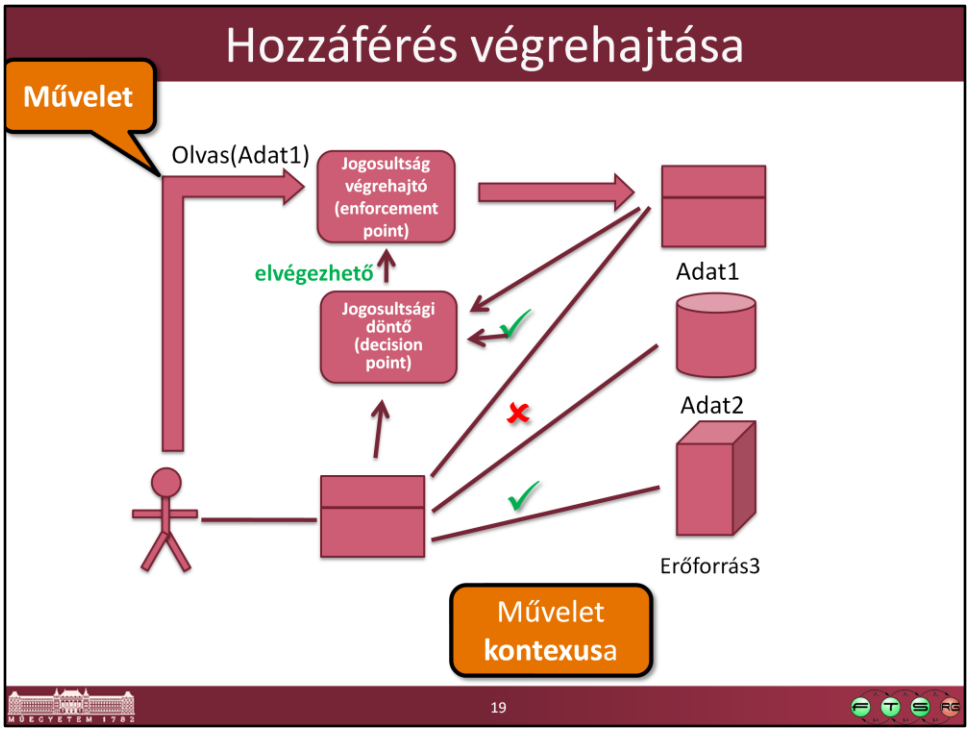
- Linux-PAM: The Linux-PAM System Administrators' Guide, URL: http://debian.securedservers.com/kernel/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html
- Kerberos: IETF. The Kerberos Network Authentication Service (V5). RFC 4120, 2005. URL: http://tools.ietf.org/html/rfc4120

Tartalom

- Számítógépes biztonság bevezető
- Felhasználók kezelése, hitelesítés
  - UNIX, Linux alatt
  - Windows alatt
- **Engedélyezés**
  - Engedélyezés általános sémái
    - Szerep alapú hozzáférés-vezérlés
    - Hozzáférés-vezérlési listák
  - Engedélyezés UNIX, Linux alatt
  - Engedélyezés Windows alatt
    - Biztonsági alrendszer alapok
    - Központosított hozzáférés-vezérlés

Engedélyezés általános sémái





- A szereplők *műveleteket* kezdeményeznek
- A műveletek *kontextusa* tartalmazza a szereplő azonosítóját, a célobjektumot és az elvégzendő művelet fajtáját
- A jogosultsági *döntő* komponens kiértékeli a kontextust, engedélyezi vagy megtiltja a műveletet
- A jogosultsági *végrehajtó* komponens biztosítja, hogy a döntő által hozott döntés érvényre jusson

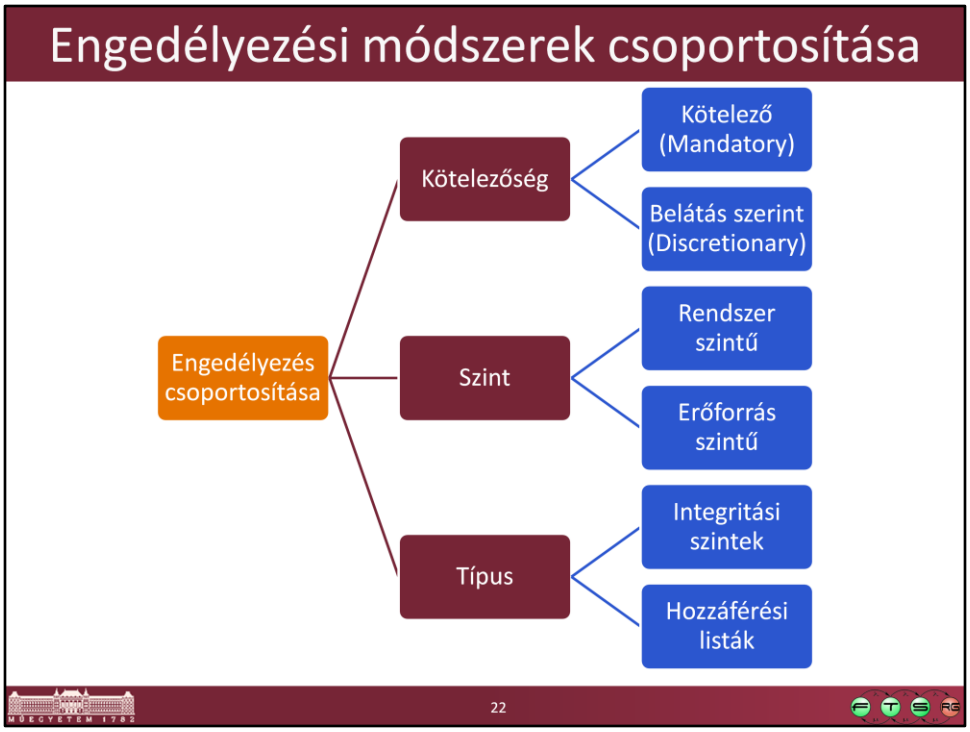
Engedélyezés gyakorlati kihívásai

- Sok szereplőt kell kezelni a rendszerben
  - Különböző rendszerek különbözőképpen azonosítják őket
- Sok védett objektumot kell kezelni
  - Különböző rendszerek ezeket is különbözőképpen azonosíthatják
- Jogosultsági relációk:
  - (Szereplők) X (Objektumok) X (Művelettípusok)
  - Az ilyet teljes *hozzáférési mátrix*nak nevezzük
  - Manuálisan (de még automatizáltan is) kezelhetetlen méretű adathalmaz

Teljes hozzáférési mátrix példa

	Objektum1	**Adat2**	**Erőforrás3**	**…**
Felhasználó1	Olvas			
Felhasználó2		Töröl, Olvas	Lefoglal	
Felhasználó3	Olvas, Ír			
Felhasználó4				
…				

- **KÉRDÉS**: mekkora lenne ez egy mai normál rendszeren?
- **CÉL:** ehelyett valami jobb adatszerkezetet és ellenőrzési módszert találni



A csoportosítás esetleges, rengeteg egyéb szempont van természetesen.

Engedélyezés fajtái - kötelezőség

- Klasszikus fogalmak (US DoD szabvány)

- Kötelező (mandatory)
  - csak központi jogosultság osztás
  - felhasználók nem módosíthatják a házirendet
- Belátás szerint (discretionary)
  - megfelelő jogú felhasználó továbboszthatja a jogokat

Engedélyezés fajtái - típus

- **Integritás védelem**
  - Címkék definiálása (+ teljes sorrendezés köztük)
  - Objektumok és szereplők címkézése
  - Kiértékelése szabályok definiálása:
    - Pl. ha nagyobb az objektum címkéje, akkor olvashatom

Integritás védelem

- Példa:
  - Címkék: `High (H) > Medium (M) > Low (L)`
  - Szereplők: `User1[H], User2[L]`
  - Objektumok: `Obj1[M], Obj2[H], Obj3[L]`
  - Mi legyen a kiértékelési szabály?

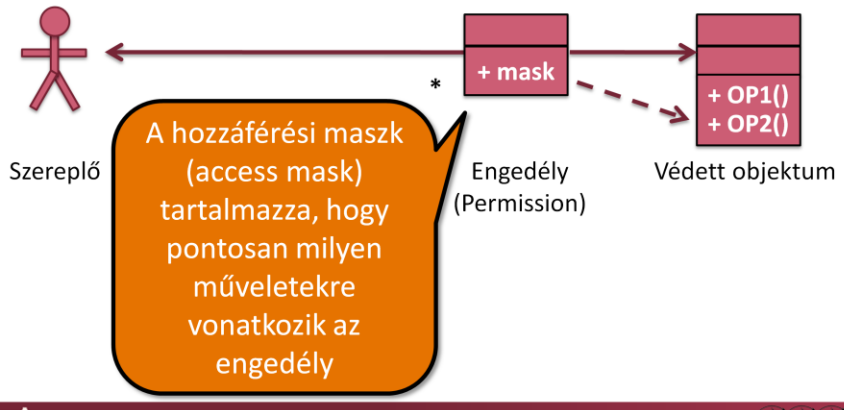
Bell LaPadula modell feltételei (bizalmasság) :
„No read up” – nem olvashatok magamnál magasabb szintű objektumból
„No write down” – nem írhatok magamnál alacsonyabb szintű objektumba

Biba modell feltételei (sértetlenség) :
„No write up” – nem írhatok magamnál magasabb szintű objektumba
„No read down” – nem olvashatok magamnál alacsonyabb szintű objektumból

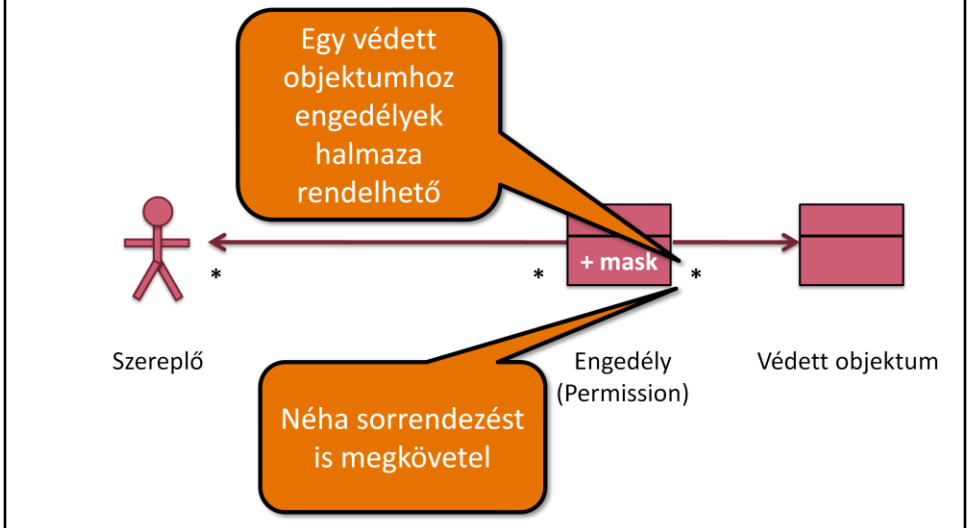
Engedélyezés fajtái - típus

- Integritás védelem
- **Hozzáférés-vezérlési listák**
  - objektum ⟶ (szereplő, engedélyek)
    - engedély: adatok írása, attribútumok olvasása…

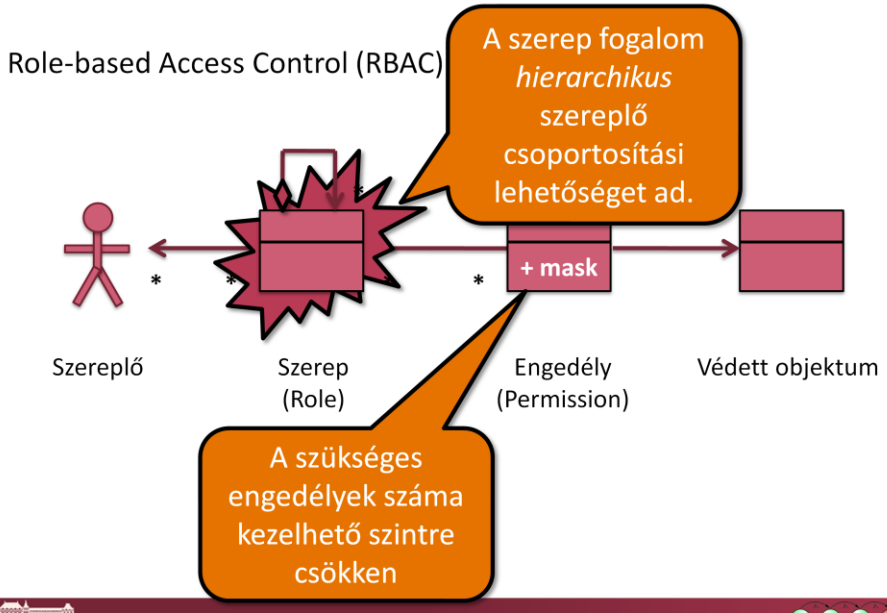
Hozzáférés-vezérlési listák



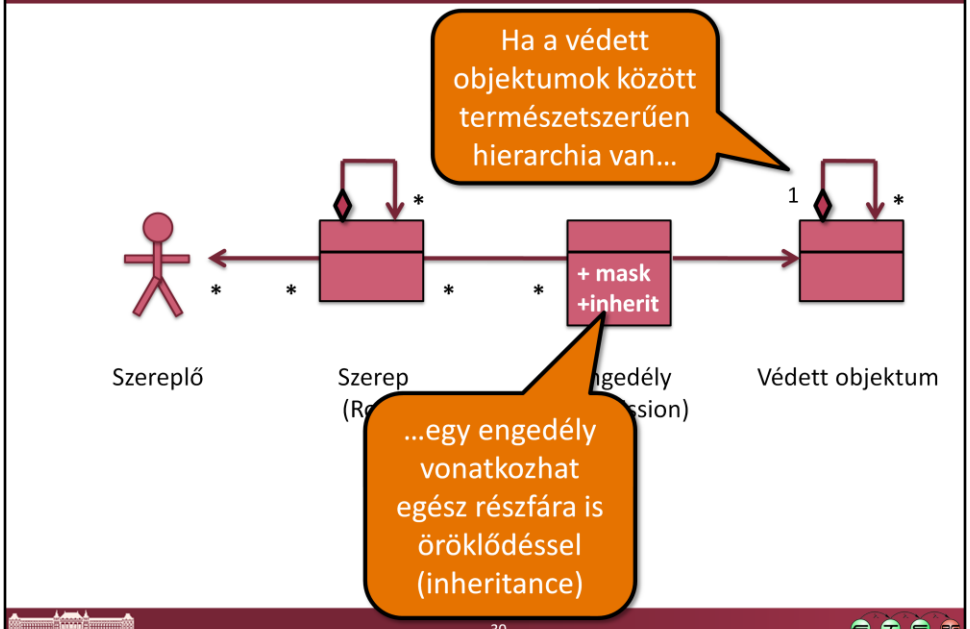
Hozzáférés-vezérlési listák



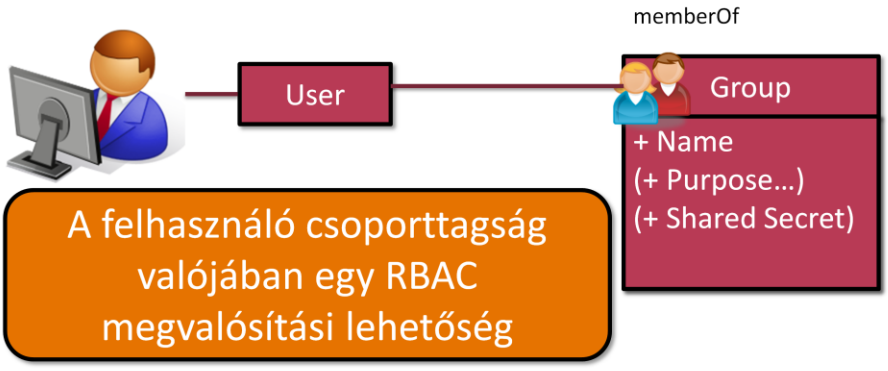
Szerep alapú hozzáférés-vezérlés



Hierarchikus objektumok



RBAC megvalósítása



A felhasználó csoporttagság valójában egy RBAC megvalósítási lehetőség

További példák:

DB szerverek; ASP.NET Role Service…

Tartalom

- Számítógépes biztonság bevezető
- Felhasználók kezelése, hitelesítés
  - Linux alatt
  - Windows alatt
  - Központosított címtárak
- Engedélyezés
  - Engedélyezés általános sémái
    - Szerep alapú hozzáférés-vezérlés
    - Hozzáférés-vezérlési listák
  - **Engedélyezés Linux alatt**
  - Engedélyezés Windows alatt
    - Biztonsági alrendszer alapok
    - Központosított hozzáférés-vezérlés, csoportházirendek

POSIX fájlrendszer jogosultságok

- Alapelemek
  - Szereplő: `user` (felhasználó)
  - Szereplő hierarchia: `group` (csoport)
  - Minden `user` teszőlegesen sok `group` tagja lehet
  - Minden `group` tetszőlegesen sok `usert` tartalmazhat
  - `Group` további `groupot` nem tartalmazhat
- Jogok
  - 3x3bit, olvasás, írás, végrehajtás (könyvtárba belépés)
    - Első a tulajdonos felhasználónak
    - Második a tulajdonos csoportnak
    - Harmadik mindenkinek
  - Speciális bitek
    - setuid, setgid: futtatásnál átveszi a file tulajdonos uid-, gid-jét,
    - sticky: újonnan létrejött fájlok tulajdonosát állítja
  - Az execute bit tiltó hatása implicit módon öröklődik a könyvtárakon, más öröklés nincs

DEMO Linux fájlrendszer jogosultságok

- Tulajdonos manipulálása: chown
  - csak rootnak engedélyezett, nem delegálható
- Jogosultság bitek módosítása: chmod
  - Csak tulajdonosnak engedélyezett
  - Többféle megadási mód:
    - Teljes felülírás 4 db oktális számmal
    - Módosítás pl.:
      - u+x (user add execute), g-w (csoport olvasás elvétele)
      - TODO: példa oktális megadás
- Listázás: ls –l illetve ls –l –n
- (POSIX ACL is létezik, idő hiányában nem tárgyaljuk)

- példa fájlok és könyvtárak létrehozása pl. /tmp/opre alatt (mkdir, touch, echo parancsok), text és shell szkript fájlok
- jogok módosítása: chmod o-r file1.txt
- más, nem root felhasználóval megnézni, nem tudja megnézni a tartalmát
- adni jogot rá, ilyenkor már megtudja
- elvenni megint, de root berakja a fájl tulajdonos csoportjába
- ilyenkor se tudja megnyitni, mert a csoporttagság a loginkor értékelődik ki (groups paranccsal lehet ellenőrizni)
- kilépni, visszalépni, és most már megy

Fájlrendszeren kívüli engedélyek

- Egyéb védett objektumok:
  - *„UNIX alatt minden erőforrás fájl”* – jó elv, sajnos nem igaz következetesen mindenre
- Sok periféria valóban rendelkezik fájlrendszerben található interfésszel
  - Pl. merevlemez teljes tartalma: /dev/sd*, soros port /dev/ttyS*
  - A kernel és teljes fizikai memória: /dev/kmem, /dev/mem
- Mi van azokkal a műveletekkel,
  - amik nem sima olvasás vagy írás jellegűek?
  - Nem kapcsolhatóak egy fájlhoz?

Fájlrendszeren kívüli engedélyek

- Speciális kiváltságok root felhasználó nevében futó folyamatoknak
  - Kérhetnek valós idejű ütemezési prioritást
  - Hozzáférhetnek közvetlenül a perifériákhoz (!)
    - Kell előtte memória illetve I/O tartomány allokáció
  - 1024 alatti TCP/UDP porton hallgathatnak
  - Kernel bizonyos konfigurációs beállításait megváltoztathatják, új modult tölthetnek be stb.
- Miért rossz ez?
  - Webszerver: 80-as porton hallgat, de nem szabadna a rendszer konfigurációját módosítania
- Nem előnyös, ha ezek nem szabályozhatóak külön-külön
  - Legkevesebb jog elve (**principle of least privileges**)
  - POSIX Capabilities mechanizmus – globális rendszerszintű erőforrásokra vonatkozó jogosultságok (ún. privilégiumok)

Capability definíciók listája: /usr/src/linux/include/linux/capability.h

Kitekintés

- Finomabb felbontású jogosultságkezelés végrehajtható fájlokra
  - Platform szintű behatolás elleni védőmechanizmusok támogatására (PAX, grsecurity)
  - A védőmechanizmusok számos egyébként sértetlen programot tesznek működésképtelenné (pl. JavaVM)
  - Speciálisan kivételezni kell az ilyen alkalmazásokat fájlrendszerbe írt címkével (SELinux Security Labels)
  - Alkalmazásokhoz hozzárendelt rendszerhívási profilok (AppArmor) – felfedi ha a „szokásoshoz” képest megváltozik az alkalmazás futása

pl. Java VM futási időben generálna végrehajtható kódot, de az NX bitre épülő „W xor X” szabály megtiltja, hogy olyan memórialap, amire ír egy folyamat később végrehajtható legyen.

Összefoglalás

- Hitelesítés és engedélyezés szétválasztása
- Engedélyezés általános fogalmai
- Megvalósítás UNIX/Linux környezetben