tech.net

working with windows

Generációváltás

IV. / 03. szám

1344 Ft

ISSN 15865185 03

9 771586 518005

Szerkesztőség:
Főszerkesztő: **Fóti Marcell**
marcellf@netacademia.net
Főszerkesztő-helyettes: **Fülöp Miklós**
mick@netacademia.net
A szerkesztőség címe:
1062 Budapest, Andrássy út 62.
Telefon: 472-1214
technet@netacademia.net
Nyilvános levelezési lista:
tech.net@technetklub.hu

Kiadja és terjeszti a
NetAcademia Kft.
Terjesztési, előfizetési információ:
Telefon: 472-1214
terjesztes@netacademia.net
Megjelenik havonta, ára 1.344 Ft

Előfizethető megrendelőlevélben a szerkesztőségnél:
1062 Budapest, Andrássy út 62.
Fax: 472-1215
http://technet.netacademia.net/subs

Hirdetésfelvétel: **Szívós Éva**
Telefon: 472-1214
Fax: 472-1215
info@netacademia.net

Nyomdai előkészítés:
Ars Luna Bt.
Vezető: Dobák Ildikó

Nyomda:
AduPrint Kiadó és Nyomda Kft.
1061 Budapest,
Paulay Ede utca 55.
Felelős vezető: **Tóth Béláné**

ISSN 1586-5185

Windows 2003 Server Expert Workshop (2 nap)

A Windows 2003 technológiai újdonságainak felfedezése.

Szakítson időt a Windows 2003 újdonságainak felfedezésére! Nálunk két nap alatt megtudhatja mindazt, amit egyedül másfél év alatt lehet összeszedni!

TÉMAKÖRÖK:

Az Active Directory újdonságai

Erdők közötti tranzitív trust; tartományok átnevezése; sémabővítés deaktiválása, felülírása; kezdeti replikáció mentésből *(DCPROMO)*; Global Catalog-mentes bejelentkezés; Active Directory Application Partition és az Active Directory-integrált DNS újdonságai; Forest és Domain Functionality Level, frissítéshez: ADPrep; LDAP-újdonságok: TLS-támogatás, TTL a bejegyzéseken, Digest autentikáció, Virtual List View, elmenthető lekérdezések; Active Directory Migration Tool, User State Migration Tool;

A Group Policy újdonságai

Resultant Set of Policy; Új házirendcsomagok: NetLogon, Credential Manager, Terminal Services, kliensoldali DNS-beállítások; Software Restriction Policies; Administrative Templates Web View; A GPO korlátozása WMI-szűréssel; Group Policy Management Console: GPO mentés-visszaállítás, export-import

Fejlesztői újdonságok rendszergazdáknak

Hasznos programocskák készítése *(Visual)* Notepad segítségével: .NET Framework; ASP.NET; ADO.NET

Rendszerújdonságok

GUID Partition Table; Headless Server; Automated System Recovery; Volume Shadow Copy; új WMI Providerek: a replikációhoz, a trust-kapcsolatokhoz, DFS-hez, nyomtatókezeléshez; Parancssori WMI *(WMIC.EXE)*

Biztonsági újdonságok

EFS: több felhasználó, titkosítás WebDAV-megosztáson
Certificate Services újdonságok: auto-enrollment felhasználók számára is, Certificate Mapping, módosítható tanúsítványsablonok, delta CRL-támogatás, központi kulcstárolás, kulcsvisszaállítás
Software Update Services

IIS 6 újdonságok

http.sys és Web Administration Service; Worker Process Isolation Mode; Web Garden Demand Start; URL Authorization; XML Metabase; POP3 és SMTP szolgáltatás

Időpontok: 2003 április 10-11.
2003 május 15-16.
2003 június 12-13.
Jelentkezés: www.netacademia.net/workshop/W2003

Nálunk mindent kipróbálhat!

Köszöntő

Generációváltás

A Microsofttól megszokott hosszas vajúdás után *(Whistler → .NET Server → Windows Server 2003)* megérkezni látszik a Windows operációs rendszer következő verziója. Mivel immár publikusan letölthető próbaváltozat is rendelkezésre áll, úgy döntöttünk, „átállunk". Ez azt jelenti, hogy mostantól mind a tech.net magazinban, mind tanfolyamainkon a Windows Server 2003 lesz az alap operációs rendszer – már ha a körülmények megengedik.

Tiltó körülmény például az Exchange 2000 jelenléte, mely olyan erősen támaszkodik a Windows 2000-ben lévő IIS-re és AD-re, hogy az új oprendszer-változat nem is jó neki.

Ha abból indulunk ki, hogy a következő Windowshoz nem kell újból letenni az összes MCP-vizsgát, azt hihetnénk, hogy nincs is benne semmilyen újdonság. De van. A **[w2003RevGuide]** címről letölthető, több mint száz oldalas Reviewer's Guide semmi mást nem tartalmaz, mint az újdonságok táblázatos felsorolását. Miközben átrágtam magam a dokumentumon, folyamatosan jegyzeteltem, mi mindennel kell majd foglalkoznunk a közeljövőben. Két A4-es oldalt tett ki az újdonságok felsorolása – címszavakban! Példaként álljon itt az Active Directory-újdonságok köre *(ezek azok, amiken van mit kipróbálni)*:

- Erdők közötti tranzitív trust; tartományok átnevezése; sémabővítés deaktiválása, felülírása; kezdeti replikáció mentésből *(DCPROMO)*; Global Catalog-mentes bejelentkezés; Active Directory Application Partition; Forest és Domain Functionality Level; LDAP-újdonságok: TLS-támogatás, TTL a bejegyzéseken, Digest autentikáció, Virtual List View, elmenthető lekérdezések; Active Directory Migration Tool, User State Migration Tool.

A teljesen újraírt IIS6 egészen új fogalomtárat vezet be, van itt WAS szolgáltatás és webkert *(webgarden)*, http.sys és XML alapú metabase. Méretes listáim vannak a Group Policy, a biztonság, a hálózatkezelés, a parancssori lehetőségek újdonságairól, az integrált .NET-keretrendszerről stb.

Jelen számunkban részletes elemzést olvashatnak a tartományok és tartományvezérlők átnevezéséről *(nem triviális!)*, néhány Group Policy-újdonságról *(folyt. köv.)*, a WMI parancssori kezeléséről *(WMIC.EXE azoknak, akik nem óhajtják fejből tudni a WMI-osztályokat)*, valamint az EFS új funkcióiról. Most pedig jöjjön egy kis önreklám:

Windows Server 2003 Expert Workshop!

Felhívom a tisztelt Olvasóközönség figyelmét, hogy a kézzelfogható ismeretek, tapasztalatok elsajátítása érdekében létrehoztunk egy kétnapos Workshopot *(lásd a szemközti oldalon)* a Windows Server 2003 újdonságainak gyakorlatban történő kipróbálására, gondolván azokra a szerencsésekre, akik nem várhatják meg, amíg lapunk hónapok hosszú során át lehozza az összes tudnivalót, hanem minél hamarabb át szeretnének térni az új rendszerre.

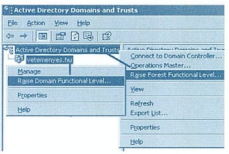
Fóti Marcell
marcellf@netacademia.net
A szerző a NetAcademia vezető oktatója
MCSE, MCT, MCDBA, MZ/X

A cikkekben szereplő URL-ekről

Talán feltűnt egyeseknek, hogy a fenti Köszöntőben a Reviewer's Guide letöltési helyére nem a korábban megszokott sorszámmal [1], hanem egy szócskával [w2003RevGuide] utaltunk. A jövőben a cikkekhez tartozó URL-eket nem nyomtatásban *(nehézkes begépelni, elavulhat stb.)*, hanem egy, a TinyURL-hez hasonló megoldással adjuk meg. A szögletes zárójelben megadott kulcsszót egyszerűen be kell írni a http://technet.netacademia.net/go? cím mögé, és már röpülünk is arra a lapra, amit a cikk szerzője szerint érdemes elolvasni.

Tehát a **[kulcsszo]** használata:
http://technet.netacademia.net/go?kulcsszo

2003. 03. szám

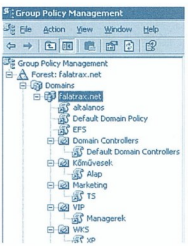


Windows Server 2003

Tartományok átnevezése

Windows 2000 esetében igen körültekintően kellett eljárni az Active Directory tartományok nevének megválasztásánál: az ugyanis a későbbiekben nem volt megváltoztatható. A Windows Server 2003 megjelenésével lehetővé vált a névcsere, bár körültekintésre most is szükség van, mert a névcsere egyben a tartományi fa átalakításának is eszköze. Ebben a cikkben végigvesszük a tartományátnevezés lépéseit, s bónuszként felvillantom az application partitionok kezelésének lehetőségeit is.

4. oldal

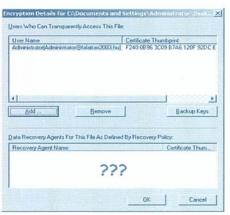


Csoportos házirendek a Windows Server 2003-ban

Bemutatkozik a Group Policy Management Console

A Windows 2000 bevezetésével elkezdett csoportházirendek vonalán is erősít a Microsoft a Windows 2003 megjelenésével. A Windows 2000-ben még újdonságnak számító csoportos házirendek kezelését teszi könnyebbé a Windows 2003 környékén megjelenő Group Policy Management Console *(GPMC)*, vagy a Resultant Set of Policy *(RSoP)*.

9. oldal

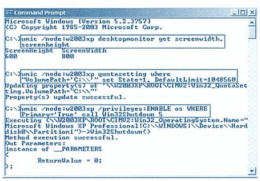


Titkosított fájlrendszer v2.0

Az Encrypting Filesystem a Windows XP-ben és a Windows Server 2003-ban

A titkosított fájlrendszer a Windows 2000-ben jelent meg. Az EFS akkori verziója még több szempontból is gyerekcipőben járt, de azóta eltelt három év és a „gyermek" szépen cseperedett. Cikkünkben bemutatjuk, mit „tud" az EFS a Windows XP-ben, illetve a Windows Server 2003 tartományi környezetben.

13. oldal



Rendszerfelügyelet parancssorból

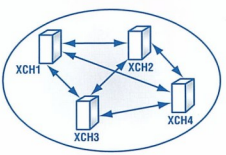
A Windows 2003 Server WMI szolgáltatásai

A WMI *(Windows Management Instrumentation)* rendszerfelügyeleti felületet 1996-ban, akkor még WBEM *(Web Bases Enterprise Management)* néven kezdte el kidolgozni a Microsoft, a Compaq, a BMC, a Cisco és az Intel. A cél egy univerzális, különféle hardver- és szoftverelemekhez egyaránt használható felület létrehozása volt. A Microsoft most még rátett egy lapáttal: a Windows 2003 Serverben egy nem is akármilyen parancssori eszközt is kaptunk hozzá.

16. oldal

A Microsoft Exchange 2000 útválasztása

Az üzenetek útja egy szervezeten belül és kívül



Egy szervezetben már két Exchange kiszolgáló esetén is felmerül a kérdés: merre mennek az elküldött üzenetek? Gyakran nem is tudja a rendszergazda, mi történik tulajdonképpen több kiszolgáló esetén. A cikkben szó lesz az üzenetek útjáról, az útválasztásról, és az összefüggésekről.

21. oldal

Eltemetett bitek: szteganográfia

Avagy nem minden arany, ami fénylik?

„...ellentétben a kriptográfiával, ahol a támadó észreveheti az üzenetet és módosíthatja azt, a szteganográfia célja, hogy a nyílt szöveget úgy rejtse el a gyanúmentes üzenetbe, hogy a támadó ne is láthassa meg, hogy a továbbított üzenet egy második üzenetet tartalmaz."

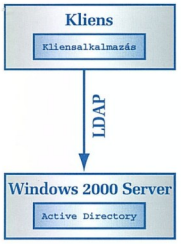
25. oldal

Portál Para[digma]

Avagy mit tegyünk, ha sürgősen portált kell építenünk?

Ha főnökeink vagy ügyfeleink azt várják tőlünk, hogy integráljunk egy-két tucat különböző alkalmazást úgy, hogy az információk és szolgáltatások az Interneten is elérhetők legyenek, mindezt egyszerű legyen kezelni, és persze az egész minimális költségvetésből és tegnapra kell, akkor bizony nehéz dolgunk van. Az ma már nem kérdés, hogy ilyenkor vállalati portált kell építeni, de hogy milyen eszközökkel, milyen architektúrával és milyen platformon, az már nem olyan egyértelmű. Jön a portál para.

30. oldal



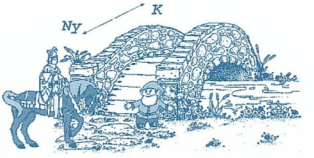
Az Active Directory programozott elérése

Sokszor esett már szó az Active Directory eléréséről, bűvöléséről, most azonban a saját kódból történő elérést szeretném bemutatni, illetve illusztrálni néhány példán keresztül. Úgy gondolom, ezek az ismeretek mind fontosak lehetnek egy AD-integrált alkalmazás fejlesztésekor.

35. oldal

Formális módszerek az informatikában [2]

A Petri-hálók további alkalmazásai



A sorozat első részeként áttekintettünk néhány alapfogalmat az informatikában használatos formális módszerekkel kapcsolatban, majd áttértünk a Petri-hálókra. A topológia bemutatása után most mélyebb részleteket szeretnék bemutatni.

39. oldal

Internet Explorer javítás

Az Internet Exploreren két újabb biztonsági rést javíthatunk a 2003. február elején megjelent MS03-004-es számú hotfixszel. Mint már megszokhattuk, ez a javítás tartalmazza az Explorerhez eddig kiadott javításokat, így ha valaki eddig elmulasztotta volna betömködni a már ismert réseket, az egyszerre megteheti az [msdownload] helyről letölthető biztonsági javítással.

43. oldal

A bitlegelők tragédiája

Egyszer volt, hol nem volt, volt egyszer sok tehénpásztor. Az egyik arra gondolt, hogy több tehén többet tejel, ezért megnövelte a csordája létszámát. Kiderült, hogy igaza lett, mire a szomszéd pásztor is hasonlóképp tett. Egy bolond százat csinál, de még a felénél sem tartottak, amikor annyi tehén bőgött a réteken, hogy már egyikük sem lakott jól teljesen...

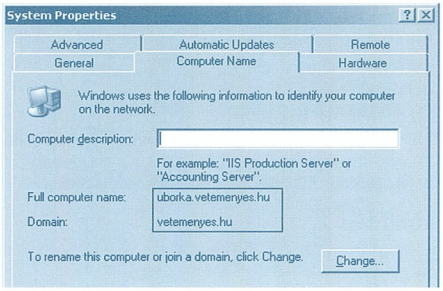
44. oldal

Windows Server 2003

Tartományok átnevezése

Windows 2000 esetében igen körültekintően kellett eljárni az Active Directory tartományok nevének megválasztásánál: az ugyanis a későbbiekben nem volt megváltoztatható. A Windows Server 2003 megjelenésével lehetővé vált a névcsere, bár körültekintésre most is szükség van, mert a névcsere egyben a tartományi fa átalakításának is eszköze. Ebben a cikkben végigvesszük a tartományátnevezés lépéseit, s bónuszként felvillantom az application partitionok kezelésének lehetőségeit is.

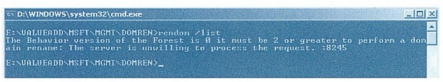
Tekintsük meg az alábbi, átnevezés előtt álló tartományt, melynek VETEMENYES.HU a neve, s a veteményesben egy UBORKA nevű tartományvezérlő csücsül *(Control Panel System ikon)*.



Uborka a veteményesben...

Az a feladatunk, hogy a tartomány neve KISKERT.HU-ra változzon. A fenti ábrán látható, csábító feliratú „Change...” nyomógomb sajnos nem alkalmas a tartomány nevének megváltoztatására, helyette a telepítő CD-n, a Valueadd\Msft\Mgmt\Domren könyvtárban található RenDom.EXE-t kell használni.

A RenDom.EXE ugyan ezernyi kapcsolóval rendelkezik, de van közöttük egy veszélytelennek tűnő változat, a /list, így nekiesünk a feladatnak, hátha azonnal sikerül... Parancssort ide!



A tartomány átnevezésének első kísérlete

Hát ez nem sikerült. Valami hiányzik. De mi?

Funkcionalitási szintek

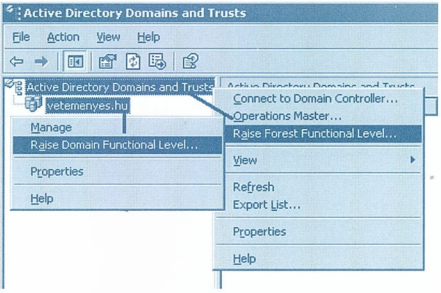
A Windows 2000-nél bevezetett Mixed és Native üzemmódok a Windows 2003-nál is megtalálhatók, csak a fogalom neve más: Functional Levelnek hívják, és külön állítható tartományi, valamint erdőszinten. A termék súgója remek összefoglaló táblázatot tartalmaz arról, hogy az egyes új AD-funkciók milyen FuncLevel meglétét igénylik.

A tartományok nevének megváltoztatásához erdőszintű 2003 FuncLevel szükséges, ami egyenlő azzal, hogy az erdő minden fájának, s abban minden tartomány minden DC-jének 2003-as Active Directoryt kell futtatnia, ráadásul 2003-as FuncLevelen.

Ez azt jelenti, hogy a névváltoztatáshoz sajnos nem elég egyetlen DC-t átállítani 2003-ra, ezt mindegyikkel meg kell tenni, majd a FuncLevelt minden tartományon, végül magán az erdőn is fel kell emelni 2003-as magasságba.

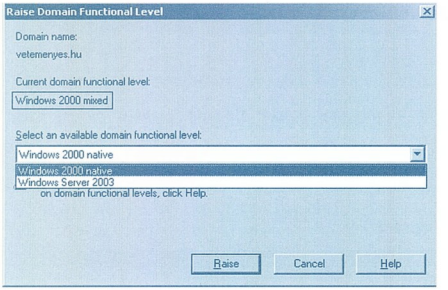
Vigyázat! A FuncLevel csak feljebb tornászható, visszaút nincs!

A funkcionalitási szint megtekintésére és átállítására az Active Directory Domains and Trusts szolgál, mégpedig úgy, hogy ha a tartomány nevén jobbkattintunk, tartományszintű, míg ha az MMC-konzol gyökerén, az erdőszintű beállításhoz jutunk, amint azt az alábbi montázs mutatja:



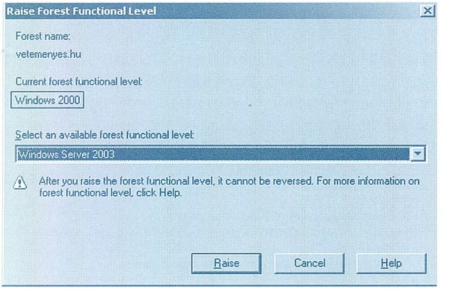
A tartományi és erdőszintű funkcionalitási szint átállítása

Először a tartományi funkcionalitási szintet kell megemelni, mert amíg létezik Mixed *(NT4-et tartalmazó)* vagy natív *(W2K-t is befogadó)* tartomány, addig az erdőszint átállítása sikertelen. A következő ábráról nemcsak az olvasható le, hogy milyen szintek léteznek, mire lehet áttérni, hanem az is, hogy a Windows 2003 AD alapértelmezésben hogyan települ: Mixed, vagyis NT4-kompatibilis üzemmódban. Annak ellenére ez történik, hogy egy különálló, friss telepítés esetén elhanyagolható annak esélye, hogy valaki NT4 Backup Domain Controllert szeretne telepíteni a tartományba. De lám, lehet.



A tartomány funkcionalitási szintjét emeljük 2003-ra!

A funkcionalitási szint felemelését az összes tartományon el kell végeznünk, mivel a cél az erdőszintű FuncLevel felemelése! Miután az összes tartománnyal végeztünk, s a replikáció is lezajlott, erdőszinten is áttérhetünk 2003-ra:



Az egész erdő funkcionalitási szintjének megemelése

Itt már csak 2000 és 2003 szintek léteznek, értelemszerűen nincs NT4-es erdőszint, mert abban a történelmi korban még nem léteztek erdők.

Az átnevezés további feltételei

1. Az új tartománynév nyilván új DNS-zónába fog tartozni, ezt az átnevezés előtt célszerű megcsinálni, és dinamikusan frissíthetővé tenni.
2. Fontos átnevezési feltétel, hogy ha az erdőben Exchange 2000 van, sajnos le kell tennünk az átnevezésről. Nem megy.
3. Tartomány átnevezésére az Enterprise Admins csoport tagjai képesek.
4. Az átnevezés során a DC-k újraindulnak, míg a RenDom.EXE-nek futnia kell, mert ő koordinálja az átnevezés menetét. Ebből az következik, hogy ne valamelyik DC-n, hanem például egy munkaállomáson futtassuk. *(Ha csak egy DC-nk van, tehát nincs szinkronizálási igény, akár azon a DC-n is futtathatjuk.)*

A RenDom.EXE

És most lássuk a „varázslót", az eszközt, amivel az átnevezési műveletet el kell végeznünk. Használata első pillantásra szokatlan, ugyanis mintegy ötször-hatszor le kell futtatnunk *(mindannyiszor más kapcsolóval)* ahhoz, hogy elvégezze nem is egyszerű feladatát.

Az eljárás menete egyébként a következő: a RenDom egy XML-fájlba exportálja a címtár átnevezhető névtereit. Ezt a fájlt valamilyen szerkesztőprogrammal *(pl. Notepad)* átírkáljuk, majd visszaimportáljuk az AD-be, ahol az replikálásra kerül. Ezután vezényszóra az összes DC nekilát a saját címtárpéldányában *(NTDS.DIT)* elvégezni az átnevezést, s ha végzett, az új néven újraindul.
A szokásos kapcsolókat *(felhasználói kontextus-váltás, egy adott géphez csatlakozás, output fájl nevének beállítása stb.)* nem részletezve az alábbi lényeges funkciói vannak a RenDom.EXE-nek *(a felsorolás sorrendje egyben a használat sorrendje is!)*:

/list	Ez a lépés kiexportálja a címtárból az átnevezhető névterek listáját a domainlist.xml nevű fájlba. *(Ezt kell majd a rendszergazdának módosítania.)* A parancs a Domain Naming Master FSMO-szerep hordozójához fordul. Ha ezt a gépet nem lehet elérni, baj van!
/showforest	Az átszerkesztett domainlist.xml alapján megjeleníti a jövőbeni AD-struktúrát. Így még az előtt le lehet ellenőrizni módosításaink helyességét, hogy belekezdenénk a műveletbe. Ha valami nem stimmel, ismét módosítsuk a fájlt!
/upload	A módosított domainlist.xml visszaimportálása a címtárba *(msDS-UpdateScript attribútum)*. Ebben a lépésben születik a dclist.xml fájl is, amely ilyenkor minden DC-ről ezt tartalmazza: <State>Initial </State> *(lásd később)*. Emellett ez a lépés elkészíti a DNS-zónába felveendő új rekordok listáját is, megpróbálja beregisztrálni az új SRV rekordokat, illetve dnsrecords.txt néven lementi az indítási könyvtárba is. Az erdőben bizonyos funkciók letiltódnak.
/prepare	Annak ellenőrzésére szolgál, hogy az összes módosítandó DC felkészült-e az egyidejű átnevezésre. Futtatásakor módosul az előző lépésben létrejött dclist.xml fájl, amiből kiolvasható a DC-k állapota. Addig nem szabad továbblépni, amíg az összes DC <State>Prepared</State> állapotba nem kerül!
/execute	Ha minden DC felkészült az átnevezésre, most vezényszóra bele is kezdenek a feladatba, majd **újraindulnak**! Vigyázat! Az újraindulás leállíthatatlan! Csak akkor indítsuk el ezt a lépést, ha nem okoz zavart a DC-k újraindulása! A folyamat végén a dclist.xml ezt tartalmazza: <State>Done</State>
/end	Sikeres átnevezés után ez a lépés visszaállítja az AD-t normális üzembe, vagyis megszünteti a /upload lépés által okozott korlátozásokat.
/clean	Ha újabb átnevezésre van szükség, előbb ezzel a paranccsal ki kell pucolni a msDS-UpdateScript attribútumot.

Talán ezzel sokakat sikerült egyszer s mindenkorra elijesztenem az átnevezéstől, pedig a lépések egyáltalán nem bonyolultak, s maga az XML-fájl is pici, aranyos.

Az első lépés

Elsőként tehát a /list kapcsolóval futtatjuk a RenDom.EXE-t, így a jelenlegi tartományneveket tartalmazó XML-fájlhoz jutunk, amit majd megfelelően át kell javítanunk.

```
1. RenDom /list
```

Az alábbi ábrán azt a domainlist.xml fájlt láthatjuk, ami a veteményeskertünkről készült *(könnyű nekem, csak egy DC-m van)*:

```
<?xml version="1.0" ?>
<Forest>
- <Domain>
    <!-- PartitionType:Application -->
    <Guid>e687de75-7e19-4b02-b055-f4883ffee801</Guid>
    <DNSname>DomainDnsZones.vetemenyes.hu</DNSname>
    <NetBiosName />
    <DcName />
  </Domain>
- <Domain>
    <!-- PartitionType:Application -->
    <Guid>48eebd84-1951-4f4c-872d-a6a28f188f49</Guid>
    <DNSname>ForestDnsZones.vetemenyes.hu</DNSname>
    <NetBiosName />
    <DcName />
  </Domain>
- <Domain>
    <!-- ForestRoot -->
    <Guid>d68ce0af-b11a-4481-93b6-d4925f45315b</Guid>
    <DNSname>vetemenyes.hu</DNSname>
    <NetBiosName>VETEMENYES</NetBiosName>
    <DcName />
  </Domain>
</Forest>
```

Az átnevezéshez használt, a RenDom által generált domainlist.xml

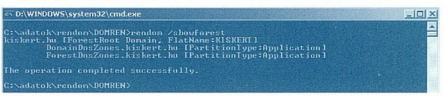
Bekereteztem azokat a részeket, ahol névcserére kerülhet sor. Két *(a DNS Server által használt)* application partition után következik a tartománynév. Itt jegyzem meg, hogy a tartomány kétféle nevét *(DNS és NetBIOS)* egymástól függetlenül meg lehet változtatni, hisz ez a két név akár eltérő is lehet. További fontos megjegyzés, hogy a DNS-név átírása általában egyenértékű a tartományi hierarchia megváltoztatásával is. Ha tehát a vetemenyes.hu helyett ezt írom: kiskert.hu, ezzel a tartományt teljesen áthelyezem a „hu" hierarchiában.
Bár a tartományátnevezés dokumentációja **[rendomdoc]** világosan leírja, hogy minden, azonos névtérbe eső nevet egyszerre kell megváltoztatni *(tehát az application partitionokét is)*, ezen valahogy átsiklott a tekintetem, így én csak magát a tartományt neveztem át, aminek hatására „zombi" AppPartitionok maradtak a címtárban. Az eset tanulságos volta miatt azonban így haladunk tovább, mert így meg tudom mutatni, hogy kell „zombikat gyilkolni".
A **[rendomdoc]** tanulmányozása egyébként is kötelező, mert mintegy ezer olyan mikrokörülményre hívja fel a figyelmet, amit egy ilyen cikkbe lehetetlen belegyömöszölni: mit tegyünk, ha SmartCard logonnal súlyosbított a helyzet, mik a lehetséges hibajelenségek, hogyan kell a GPO-linkeket az átnevezés után frissíteni stb.
Most ezeket a finomságokat hagyjuk, gondolatban írjuk át a fenti XML-fájlban a vastagon jelzett tartományneveket VETEMENYES-ről KISKERT-re.

A módosítások ellenőrzése

Mielőtt továbbmennénk – különösen összetett tartományfák esetén – érdemes leellenőrizni a változtatásokat. A

```
2. RenDom /showforest
```

segítségével „grafikus" formában meg tudjuk tekinteni, mivé is alakulna az erdő, ha ezzel az XML-fájllal tovább dolgoznánk:



A módosítások utáni AD-erdő „grafikus" megjelenítése

Ha a struktúra megegyezik azzal, ami a módosításunk célja, jöhet a következő lépés, az XML-fájl visszatöltése az AD-ba.

Vigyázz, kész…

```
3. RenDom /upload
```

Mindössze ennyit kell tennünk, és a módosítások *(maga az XML fájl!)* bekerülnek az msDS-UpdateScript attribútumba *(Configuration naming context, Partitions konténer)*.
Nagyon fontos tudnivaló, hogy amerre az msDS-UpdateScript attribútum szétreplikálódik, a tartományi erdő zárolás alá kerül. Amíg véget nem ér az átnevezés, nem lehet:

- új DC-t telepíteni,
- új tartományt beilleszteni az erdőbe,
- trust-kapcsolatokat készíteni!

Ezek a korlátozások majd a /end kapcsolóval szüntethetők meg, nyilván a teljes folyamat sikeres lezárulása után.
A parancs futtatására válaszként ismét egy XML-fájlt *(dclist.xml)* kapunk, ebben találjuk leírva, hogy melyik DC hol tart az átnevezési folyamatban:

```
<?xml version="1.0" ?>
- <DcList>
    <Hash>5OGUgza0d0iNddYPyLaViopcU+s=</Hash>
    <Signature>yOBKlxm+Im0V4zv6rKFdJlNDt6E=</Signature>
  - <DC>
      <Name>uborka.vetemenyes.hu</Name>
      <State>Initial</State>
      <Password>SfwPznrMp4w=</Password>
      <LastError>0</LastError>
      <LastErrorMsg />
      <FatalErrorMsg />
      <Retry />
    </DC>
  </DcList>
```

A kocka el van vetve, az átnevezés elindult! (dclist.xml)

Most egy adag várakozás következik, hisz meg kell várnunk, amíg:

- elkészülnek az új SRV-rekordok a DNS-ben *(amelyek egyébként a régi hostrekordokra mutatnak!!!)*
- az msDS-UpdateScript attribútum szétreplikálódik az erdőben

Miért van az, hogy az új SRV-rekordok a régi hostnevekre fognak mutatni? Kinek kell ez a kavarás? A munkaállomásoknak! Az átnevezés során a tartomány összes gépének DNS-neve *(FQDN)* magától átáll majd az új névre, azaz a hostrekord átköltözik az új zónába – kivéve a tartományvezérlőket!
Éppen ezért a korábbi DNS-zóna *(vetemenyes.hu)* törlését nem szabad megtenni addig, amíg a DC-k teljes DNS-nevét egy kétlépéses folyamat során *(NetDom.EXE)* kézzel át nem állítgatjuk!
Ez azért van így, hogy ne legyen olyan pillanat, amikor a DC-k megtalálhatatlanok, „elugranak" a munkaállomások elől. A tartományvezérlők FQDN-jének megváltoztatását lásd később.

A felkészülés ellenőrzése

Futtassuk a

```
4. RenDom /prepare
```

parancsot, ami frissíti az előbb mutatott dclist.xml fájlt. Addig kell várnunk, amíg az összes DC <State>Prepared</State> állapotba kerül.
Esetleg próbálkozhatunk azzal, hogy az Active Directory Sites and Services segítségével felgyorsítjuk a replikációt, mindenesetre idővel minden DC felkészülten várja a nagy pillanatot. Ha ez mégsem következik be, az átnevezési folyamatot megszakíthatjuk a /end kapcsolóval.
A mi esetünkben minden *(mind az egy ;)* DC felkészült állapotba került.

...rajt!

```
5. RenDom /execute
```

Emlékeztetőül: ez a lépés az összes érintett DC-n megszakíthatatlan újraindulást fog okozni, csodálkoztam is nagyon, amikor legelőször csináltam!
Tulajdonképpen itt van jelentősége annak, hogy ne a DC-k valamelyikén futtassuk a programocskát, mert ha a gép újrabootol alatta, bajosan fogja tudni frissíteni a dclist.xml fájlt, pedig ez alapján tudjuk eldönteni, mindenütt sikeres volt-e az átnevezés. Ha végül <State>Done</State> állapotba kerülnek, akkor jó. Ha <State>Error</State> lesz a vége, forduljunk bizalommal a Microsoft Tudásbázishoz.

A folyamat befejezése

Végül szabadítsuk meg az Active Directoryt láncaitól, hogy ismét minden funkció *(új DC, új tartomány, új trust)* elvégezhető legyen az erdőben:

```
6. RenDom /end
```

A parancs hatására kipucolódik az msDS-UpdateScript attribútum, s ennek elterjedésével fokozatosan felszabadul az AD-erdő. Ezután következik a munka dandárja *(a teljes teendőlistát lásd a **[rendomdoc]**-ban)*!

- munkaállomások és tagkiszolgálók ***kétszeri(!)*** újraindítása
- új számítógéptanúsítványok kérése
- a Start Menü ikonjainak kijavítása a DC-ken *(például elromlik a Domain Security Policy)*
- régi DNS-zónafájlok törlése *(ezzel még várjunk!!!)*
- stb.

Group Policy linkek kijavítása *(GPFixUp)*

A csoportos házirend olyan objektum, melynek fele az AD-ban található *(jogosultságlisták, a GPO neve stb.)*, fele pedig fájlokban *(templatek stb.)*. Ezek kölcsönösen hivatkoznak egymásra, s minden megy, mint a karikacsapás mindaddig, amíg a hivatkozások élnek. Az AD-ból fájlba mutogatás nem romlik el a tartomány átnevezésének következtében, mert GUID-nevű könyvtárakra mutogatunk, s a GUID nem változik.
A visszirányú mutogatás azonban elromlik, a GPO-fájlok ugyanis LDAP-nevekkel mutogatnak vissza a címtárra. Az átnevezés után tehát az összes policyfájlon végig kell futni, és a benne lévő tartományhivatkozásokat ki kell javítani.
Ezt a RenDom.EXE mellett található GPFIXUP programocskával kell elvégezni:

```
gpfixup /olddns:vetemenyes.hu
    /newdns:kiskert.hu
    /oldnb:VETEMENYES
    /newnb:KISKERT
    /dc:CUKKINI
```

A paraméterezés szerintem önmagáért beszél. A /dc kapcsolóval – a GPO-szerkesztés ősi szabályainak értelmében – a PDC Emulatorra mutassunk! Ott szépen kijavulnak a házirendfájlok, s a File Replication Service segítségével eljutnak a többi tartományvezérlőre is.

A tartományvezérlők átnevezése

Mint azt a bevezetőben, valamint a szemközti hasábban kifejtettem, a DC-k teljes domainneve *(FQDN)* nem áll át az új névre. Ezt a lépést nekünk kell megtennünk a NetDom.EXE segítségével, aki régi cimboránk: már az NT4 Resource Kitnek is része volt *(ma pedig a telepítőlemezen csücsül a Support\Tools könyvtárban)*. Akkoriban FQDN-átirkálásra még nem lehetett használni, de az összes korabeli tartományfunkcióra igen: munkaállomás csatlakoztatására, trust létrehozására stb.
Az új változat lesz alkalmas az FQND átállítására, mégpedig oly módon, hogy lehetővé teszi, hogy egy adott gépnek **egy időben két FQDN-je legyen!** Ez tehát a titok nyitja, így biztosan nem fogunk elveszíteni egyetlen munkaállomást sem!
Ha már FQDN-csere, egy nagy lélegzettel az UBORKA nevet is kicserélhetjük CUKKINI-re, így:

További FQDN felvétele a gépre

Mint az látható, egyértelműen azt a válasz kapjuk, hogy új gépnév **added,** vagyis valahol a réginek is meg kellett maradnia. Meg is maradt. Ha most a /enumerate kapcsolóval kilistázzuk tartományvezérlőnk jelenlegi neveit, ezt kapjuk:

Csoda Piripócson: két FQDN egy gépen!

Két lépés van már csak hátra: a két FQDN sorrendjének felcserélése *(ami újraindítást kíván)*:

```
netdom.exe computername uborka
 /makeprimary:cukkini.kiskert.hu
```

Valamint *(reboot, és az új hostrekord meglétének ellenőrzése után)* a korábbi név eltávolítása:

```
netdom.exe computername cukkini
 /remove:uborka.vetemenyes.hu
```

Megfigyelhető, hogy ebben a legutóbbi parancsban már az új gépnévre kellett hivatkoznom a csatlakozáskor!

Újbóli átnevezés előtt...

Ha valaki ezek után notórius tartomány-átnevezővé válik, részvétem, de ehhez a mániához még tudnia kell azt is, hogy újabb tartományátnevezés előtt le kell futtatni a

```
7. RenDom /clean
```

parancsot!

„Röviden" ennyi. Az elnagyolt részeket csak a Windows Server 2003 Expert Workshop tanfolyamon mutatjuk be, mert azok esetében *(is)* sokkal többet ér a gyakorlat, mint a sok leírt betű.

BONUS: Application Partitionok kezelése

„Szerencsésnek" mondhatom magam, mert a művelet során elkövettem egy óriási baklövést, nevezetesen a DNS által használt application partitionöket nem neveztettem át *(emlékezzünk: domainlist.xml fálj szerkesztése)*. Mivel azonban a DNS névtér is megváltozott, a DNS Server kiköltözött az általa addig használt partíciókból, amelyek üresen álltak a címtárban – hátha beleköltözik valaki.
Erről NTDSUTIL segítségével győződtem meg, az alábbiak szerint *(és ez egyben az application partitionok kezelésének hivatalos módja is)*:

```
ntdsutil
```

Erre megjelenik az ndsutil parancssora. Itt adjuk ki a

```
domain management
```

parancsot. Ezzel belépünk a tartománykezelési ágba, ahol a műveletek megkezdése előtt meg kell határoznunk, melyik DC-vel, és milyen kontextusban kívánunk dolgozni. Ezért bemegyünk a

```
connection
```

almenübe, ahol megadhatjuk a kezelni kívánt DC nevét

```
connect to server CUKKINI
```

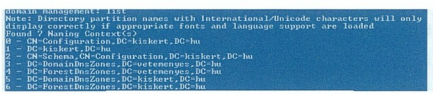
innen egyet feljebb kell lépnünk, hogy ismét a domain management szintjén legyünk, tehát

```
quit
```

Ha ez a sok hókuszpókusz mind megvolt, a

```
list
```

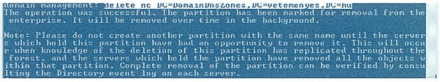
paranccsal kilistázhatjuk az Active Directoryban megtalálható névtereket:



Az AD által kezelt névterek (Naming Context)

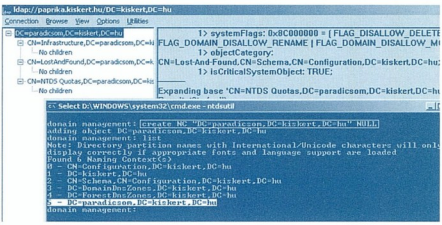
Itt látható a Configuration, a Default *(DC=kiskert,DC=hu)*, a séma, valamint a 2003-ban már felhasználói alkalmazás által is létrehozható application partitionök. Az ábráról jól leolvasható, hogy két-két DomainDnsZones és ForestDnsZones is létezik, ezek közük a vetemenyes.hu nevűek egyszerűen feleslegesek. Nem is maradtak volna meg, ha a domainlist.xml fájlt minden szükséges ponton módosítottam volna az átnevezés során, de sajnos előbb járt a kezem, mint az eszem, így ottmaradtak. Most mit tehetünk?

Egyrészt örök időkig otthagyhatjuk, mert nem sok vizet zavarnak, másrészt ki is törölhetjük őket – mert nem sok vizet zavarnak. Lássuk a törlést. Figyelem! Active Directory Application Partition-törlés következik! Halk dobpergést kérek!



Naming Context törlése az AD-ból

Hát ez sikerült. Az utasítást megismételhetjük a ForestDnsZones névtérre is, az is el fog tűnni.
Ennél azonban fontosabb, hogy olyan helyen állunk az NTDSUTIL-ban, ahonnan új application partitionok létrehozására nyílik lehetőség! Készítsünk egy PARADICSOM nevű alkalmazáspartíciót!



Új alkalmazáspartíció készítése

Hogy ez mire jó? Erről már egy programozót kellene megkérdezni.

Fóti Marcell
marcellf@netacademia.net
A szerző a NetAcademia vezető oktatója
MCSE, MCT, MCDBA, MZ/X

Tartományátnevezést jó pénzért vállalok!

Kapcsolódó tanfolyamaink:
Windows 2003 Server Expert Workshop

Csoportos házirendek a Windows Server 2003-ban

Bemutatkozik a Group Policy Management Console

A Windows 2000 bevezetésével elkezdett csoportházirendek vonalán is erősít a Microsoft a Windows 2003 megjelenésével. A Windows 2000-ben még újdonságnak számító csoportos házirendek kezelését teszi könnyebbé a Windows 2003 környékén megjelenő Group Policy Management Console *(GPMC)*, vagy a Resultant Set of Policy *(RSoP)*.

Újdonságok az ügyfélmenedzsment területén

A Windows 2003-ban számos újdonságot illetve még több finomítást találhatunk az ügyfélmenedzsment területén; eddig nem megoldott, vagy hiányzó funkciók kerültek a Windows Server 2003-ba. Íme néhány közülük:

- A csoportos házirendek újdonságai közül az egyik leglátványosabb a Group Policy Management Console *(GPMC)*, amely jelentősen megkönnyíti a házirendek mindennapi kezelését, a hibaelhárítást.
- A Resultant Set of Policy segítségével előre láthatjuk a házirendek hatásait, és utólag az összes beállítást tudjuk ellenőrizni.
- WMI szűrők segítségével egyedi módon lehet meghatározni a házirendek hatáskörét.
- Az adminisztratív sablonokban számos újdonsággal találkozhatunk, így például a terminálkiszolgálók, a felhasználói profilok, a DNS vagy az alkalmazás-kompatibilitás beállításait is szabályozhatjuk a házirendből.
- Végre igazán lehet majd korlátozni a felhasználókat a Software Restriction Policy segítségével: milyen alkalmazásokat futtathatnak és melyeket nem? Nehéz lesz kibúvót találniuk, ha például a futtatható állományból képzett HASH alapján szabályozzuk a futtatható alkalmazások körét!
- Egyre több parancssoros program segíti a rendszergazdák munkáját. A Microsoft arra törekedett, hogy helyből lehessen használni őket, de a biztonság kedvéért a súgót is megerősítették. A parancssori eszközöket távoli menedzsmentre is használhatjuk *(/S kapcsolóval.)*
- WMIC is a rendszergazdák munkáját hivatott segíteni, egyszerűbbé téve a scritpek futtatását. Erről az eszközről külön cikk szól jelen lapszámunkban.
- A RIS-t is javítgatták, megújították Redmondban. Ezentúl a kiszolgálótermékeket is lehet majd RIS-sel telepíteni. *(Eddig is lehetett egy részüket, de a megoldás kézi beavatkozást igényelt).* Ezen kívül a RIS kliens telepítővarázslóját – vagyis a CIW képernyőket – is lehet automatizálni, így még kevesebb beavatkozás szükséges a telepítések elindításához.

A sok újdonság közül először a Group Policy Management Console-t szeretnénk bemutatni. Íme:

Group Policy Management Console *(GPMC)*

A rengetegben az egyik leglátványosabb újdonság a GPMC. Egy újabb Management Console modul, amely egy átfogó program házirendet használó rendszergazdák számára. Átláthatóságot biztosít a házirendek háza táján, eddig megoldhatatlannak tűnő, vagy csak keservesen megoldható feladatok elvégzéséhez is használható.

Telepítése

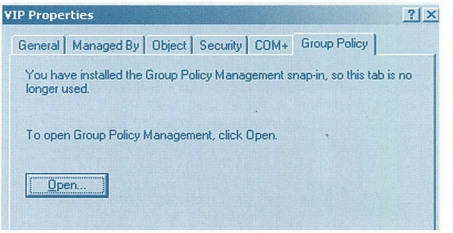
A cikkírás idején a GPMC Beta 2 állapotban van, a végleges verzió várhatóan a Windows 2003 megjelenésekor lesz elérhető.

A Beta 2 a következő környezetekben telepíthető:

- Windows Server 2003 vagy
- Windows XP SP1 + Microsoft .NET Framework + QFE Q326469 javítócsomag

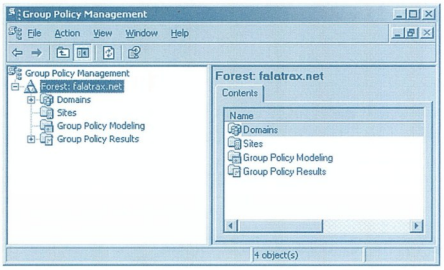
Jelenleg a windowsbeta.microsoft.com lapról tölthető le, gpmc.msi telepítőcsomagként. A telepítés roppant egyszerű, az MSI csomag futtatásának végeredményeként a GPMC a \Program Files\gpmc könyvtárba költözik. MMC modulként érhető el, és az Administrative Tools menüben is megjelenik, Group Policy Management néven.

Ha Windows 2003 kiszolgálón telepítjük, mellékhatásként a telepítés után már csak ezzel az eszközzel tudjuk a házirendeket állítgatni. Ilyenkor a Windows 2000-nél megszokott helyen, például az Active Directory Users and Computersben egy konténeren a Group Policy lapra kattintunk, már csak az alábbi üzenet vár:



A Group Policy Management átveszi a hatalmat

A Group Policy Management Console telepítése után a rendszergazda már nem nagyon használhatja az addigi házirendkezelési eszközöket az adott számítógépről. De nem kell megijedni, nem csak a Windows 2003-as, hanem a Windows 2000-es házirendeket is kezeli a Management Console, igaz, az első esetben több mindenre használható. De ne szaladjuk ennyire előre, előbb ismerkedjünk a felülettel!



Az első indítás utáni kép egy Windows 2003 kiszolgálón

A képen a falatrax.net erdő csoportos házirend menedzsmentje látható. A **Domains** konténer alatt – értelemszerűen – az erdőt alkotó tartományok találhatók. Elárulom: ebben az esetben egyetlen tartományból áll az erdő. Egyébként a rendszergazda szabályozhatja, mely tartományok látszódjanak a konzolból. A Domains feliraton állva a gyorsmenüből *(jobbkatt az egérrel)* **Show Domains** paranccsal lehet felvenni a tartományokat a konzolba. Felhívnám a figyelmet, hogy több tartomány esetén ebből a konzolból magát a tartományhierarchiát nem láthatjuk, a tartományok függetlenül kezelhetők. Ez abból adódik, hogy a csoportos házirendek valódi alapja, egysége egy tartomány. A házirendek eddig és ezután sem öröklődnek tartományok közt.

A Sites konténerben az erőhöz tartozó telephelyeket láthatjuk. Szintén a rendszergazda határozza meg, mely telephelyeket szeretné látni a konzolban, és melyeket nem. A megjelenítés/elrejtés módszere is hasonló, a Sites konténeren állva jobb klikk Show Sites paranccsal lehet kiválasztani a megjelenítendő telephelyeket. Ha kibontjuk a telephelyeket, láthatjuk a hozzájuk kötött házirendeket.

Megjegyzem, a gyorsmenüt minden konténeren állva érdemes végignézni. Többek közt gyorsmenüből lehet indítani más menedzsment eszközöket is, például az Active Directory Sites and Servicest vagy az Active Directory Users and Computerst.

Még mindig a második ábránál maradva – Windows 2003-as Active Directory esetén – látható két funkció, amiről most röviden írok csupán.

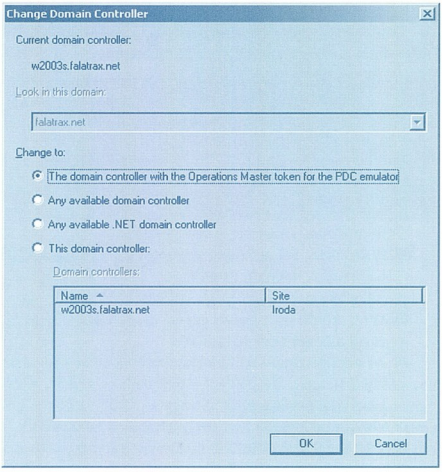
Az egyik a **Group Policy Modelling** – ami megfelel a Windows 2003-ban megjelenő új funkciónak, a Resultant Set of Policy tervezőmódjának *(RSoP planning mode)*. Segítségével előre láthatjuk, milyen hatásai lesznek a tervezett házirendbeállításoknak.

A másik a **Group Policy Results** – ami megfelel a Resultant Set of Policy naplózómódjának *(RSoP logging mode)*. Ezzel konkrét számítógép-felhasználó párosra vonatkozóan tudjuk megnézni az éppen érvényben levő házirendek pontos beállításait, azaz a végeredményt.

Nem mintha eddig nem lehetett volna kideríteni a pontos végeredményt *(a Windows 2000 Resource Kit gpresult és gpotool programok segítségével)*, de az RSoP lerövidítheti a hibakeresés idejét, használata könnyebb, grafikus felülete van, és *(tervezőmódban)* előre is láthatunk vele, bár ez csak Windows XP-vel és Windows 2003-mal működik. Úgy gondolom, ez a témakör külön megér egy cikket, ezért itt most nem mennék bele további részletekbe, maradjunk a Group Policy Management Console-nál.

Tartományvezérlők és a GPMC

Talán emlékszik a kedves olvasó, hogy Windows 2000 Active Directory esetén a csoportos házirendeket alapértelmezésben arról a tartományvezérlőről nyithatja meg a rendszergazda, amelyik a PDC-emulátori funkciókat látja el. Nos az alapértelmezés most sem változott. A GPMC is a tartomány*(ok)* PDC-emulátorához fordul. De csakúgy, mint korábban, most is átálítható, mely tartományvezérlőhöz csatlakozzon a konzol.



A tartományvezérlő kiválasztása a tartomány nevén állva a gyors menüből **Change Domain Controller** paranccsal történhet. Több telephelyes környezetben mindenképp érdemes szabályozni, hova is fordul a konzol, hisz a PDC-emulátor lehet egészen messze is.

Több erdő kezelése egy konzolból

Nem csak egy tartomány házirendjeit lehet elérni a konzolról, hanem az Active Directory erdőben *(forest)* levő valamennyit. Sőt! A képen ugyan nem látszik, de több erdő csoportos házirendjét is képes a rendszergazda egyetlen felületről a GPMC segítségével kézben tartani! Vegyesen lehet használni Windows 2000 és Windows 2003 erdők házirendjeinek kezelésére! *(Ahhoz, hogy más erdők házirendjei is elérhetők legyenek csupán egy kétirányú trustot szükséges kiépíteni a két erdő közé a Beta 2 állapotban. A végleges programban várhatóan elég lesz egyirányú megbízott kapcsolat a működéshez, ezt majd a rendszergazda szabályozhatja.)*

Ha valaki már most arra vágyik, hogy a Windows 2003-as környezetéből menedzselje a Windows 2000 Active Directoryt, és mindehhez nem szeretne kétirányú trustot, a regisztrációs adatbázisban kis változtatás után már a Beta 2-es állapotú

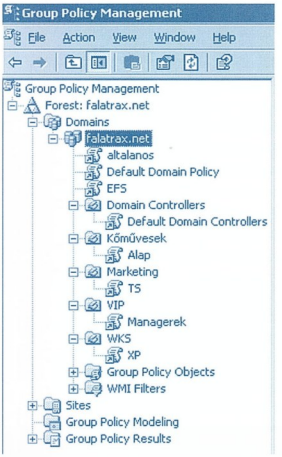
GPMC-t is használhatja. Mindehhez a következő registrymódosítást kell végrehajtani:

```
HKEY_CURRENT_USER\Software\Microsoft\Group Policy
Management console
AssumeForestTrust = 1
```

A beállítások érvényre juttatásához registrymódosítás után a GPMC-t újra kell indítani. A végleges verzióban az ígéretek szerint a felhasználói felületről lehet majd beállítani ezt a funkciót.

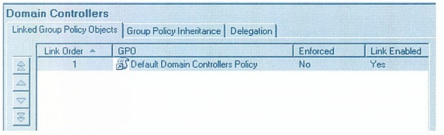
Egy tartomány kezelése

Szűkítsük a kört és nézzük, mit találunk egy tartomány keretein belül a konzolban!
Bal oldalon látható az összes olyan konténer, amelyhez házirendet tudunk rendelni, valamint azoknak a házirendeknek a listája, amelyek közvetlen a tartományhoz tartoznak.



Az ablak BAL oldala

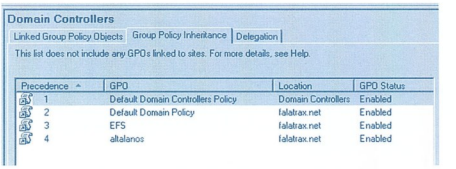
Egy konténeren állva a jobb oldalon rögtön megjelennek az adott helyhez közvetlenül kötött *(linkelt)* házirendek.



Ahogy a képen is látszik, a Domain Controllers konténerhez egyetlen házirend kötődik közvetlenül, az alapértelmezett Default Domain Controller Policy.

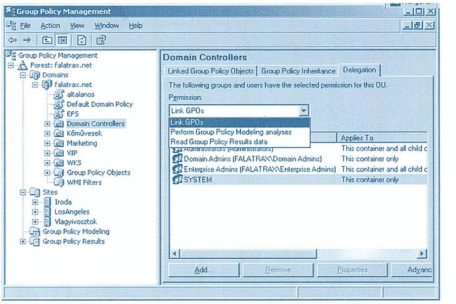
Házirendek öröklődése

Ezen a lapon láthatjuk az adott konténerre ható összes házirendet, beleértve azokat is, amelyeket örökölt egy konténer a hierarchia felette levő részeiből. Vigyázat! Ebben az ablakban nem láthatóak a telephelyekhez kötött házirendek!



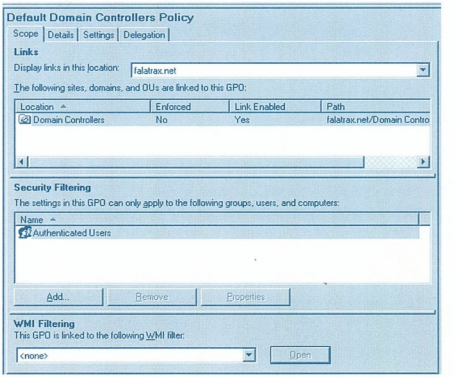
A Domain Controllers konténerre ható összes házirend

A Group Policy Inheritance fülön tipikusan olyan beállításokkal találkozunk, amelyek a házirendek öröklődésével kapcsolatosak és befolyásolják a házirendek hatását. Rögtön az első – Precedence feliratú – oszlopban látható például a hatályos házirendek prioritási sorrendje. De látható az is, hogy mely házirendet honnan örökölte a konténer *(Location oszlop)*.
A hatáskört befolyásolja, ha egy konténeren megakadályozzuk a házirendek öröklődését *(Block Inheritance)*, illetve ennek ellentéte az Enforcement *(eddig No Override néven találkoztunk vele)*, amely továbbra is erősebb a Block Inheritance beállításnál. Még mindig ugyanazon a konténeren, de jobb oldalt a Delegation lapon állva láthatjuk a jogosultságokat az adott konténerhez.



A Domain Controllers konténer Delegation lapja.

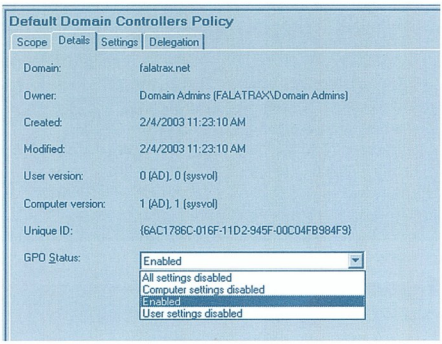
Ha bal oldalon kiválasztunk egy házirendet, akkor a jobb oldalon annak a részletei láthatóak. Az alábbi képen a jobb oldal látható.



Default Domain Controllers házirend részletei – Scope lap

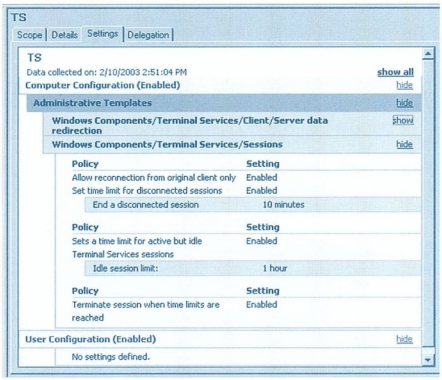
Látható rögtön, hogy kire érvényes a házirend. Ugyanezen a lapon alul állíthatók a WMI szűrők, amik egyedi tulajdonságok alapján teszik lehetővé a hatásterület pontosabb meghatározását. Erről később részletes is lesz szó.

A Details fülön – ami az alábbi képen is látható – kaphatunk információt a házirendről, például leolvashatjuk a házirend GUID-ját, vagy azt, hogy felhasználókra, számítógépekre vagy mindkettőre tartalmaz beállításokat, vagy esetleg le van tiltva.



Egy házirendről kapható információk és beállítások

A Settings fülön – ami az alábbi képen is látható – részletesen megnézhető a házirend összes beállítása érthető, könnyen emészthető formában.



HTML alapú riport a TS házirendről

A fenti Settings fülre kattintva tulajdonképp egy HTML riport jön létre az adott házirendről. Bárki, akinek olvasási joga van az adott házirendhez, képes ilyen riportot létrehozni. Korábban, ha egy felhasználónak vagy rendszergazdának nem volt olvasási és írási joga egy házirendhez, meg sem tudta nyitni a Group Policy Editorral.

A GPMC-vel innen már csak egy lépés, és HTML vagy XML fájlba exportálhatjuk az összes beállítást, jelentősen megkönnyítve ezzel a dokumentálást. Az exportált fájlban nemcsak a beállítások, de a Scope, Details és a Delegation fülek tartalma is szerepel.

Az adminisztratív sablonok és a GPMC

A riport létrehozásához a GPMC az adminisztratív sablonokból veszi a megjelenített neveket. Ha – mondjuk – magyar nyelvű riportot szeretnénk, használjunk magyar sablonokat hozzá! Meghatározható, honnan vegye fel a sablonokat a GPMC a riportkészítéshez.

A GPMC a következő sorrendben jár el.

1. Megnézi, a következő registry kulcs értékét:

```
HKEY_CURRENT_USER\Software\Microsoft\Group Policy
Management Console
ADMFilePath (REG_SZ)
```

Ha ez ki van töltve, innen veszi a sablonokat. Érdemes tehát beállítani.

2. Ha a registry kulcs értéke nincs szabályozva, körülnéz a %windir%\inf könyvtárban, ha ott talál bármit, az alapján készíti a riportot.
3. Ha az előbbi két helyen nem jár sikerrel, a házirend SYSVOL alatt levő könyvtárából olvassa be a sablonokat.

A GPMC nem nézi végig az összes helyet: ha az egyikben már talált sablonokat, azokat használja. Alapértelmezésben például a registry kulcs nincs szabályozva, viszont a %windir%\inf könyvtárban talál sablonokat, akkor annak segítségével képzi a riportot, a 3. helyet már meg sem nézi. Jön a kérdés, mi történik akkor, ha nem a házirendhez tartozó sablonok alapján készül a riport? Ebben az esetben minden beállítás, amihez a sablonokban nem talál magyarázatot, extra beállításként jeleníti meg. A riportban közvetlenül a registry kulcs és értéke jelenik meg. Érdemes tehát odafigyelni honnan is szedi a GPMC a sablonokat, mert könnyen furcsa riportokat kaphatunk.

Folytatjuk...

Dorner Csilla
dorner.csilla@netacademia.net

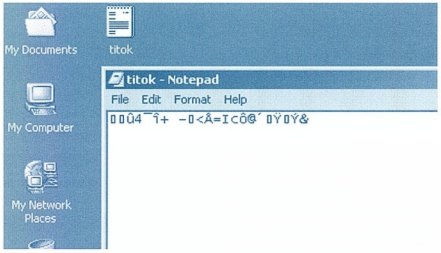
Titkosított fájlrendszer v2.0

Az Encrypting Filesystem a Windows XP-ben és a Windows Server 2003-ban

A titkosított fájlrendszer a Windows 2000-ben jelent meg, mint új, üdvözlendő módszer adataink biztonságos tárolására. Az EFS akkori verziója még több szempontból is gyerekcipőben járt, de azóta eltelt három év és a „gyermek" szépen cseperedett. Cikkünkben bemutatjuk, mit „tud" az EFS a Windows XP-ben, illetve a Windows Server 2003 tartományi környezetben.

Algoritmusok

Az EFS titkosító algoritmusa a Windows 2000-ben kötelezően a DESX volt. A Windows XP SP1 *(!)* és a Windows Server 2003 *(mostantól egyszerűen csak „W2K3")* megjelenésével azonban ennek helyébe az amerikai kormányzati körökben DES-t felváltó új titkosítási algoritmus, az AES *(Advanced Encryption Standard, „gyermekkori" nevén Rijndael)* lép. Bár az AES különböző hosszúságú kulcsokkal képes üzemelni, a Windows minden esetben 256 bites kulcsokat használ a titkosításhoz. Az AES mellett használható algoritmusként megjelent még a 3DES is, de az alapértelmezés minden esetben az AES marad. Nagyon fontos, hogy a Windows 2000 csak a DESX algoritmust ismeri, ezért az AES *(sőt, akár a 3DES)* algoritmussal titkosított fájlokat a Windows 2000 nem képes helyreállítani.



Ez nem sikerült: így néz ki egy AES-sel titkosított fájl, miután a Windows 2000 DESX-ként dekódolta

Az XP SP1 és a W2K3 képes dekódolni a DESX, 3DES és persze az AES algoritmusokkal is. A Windows XP SP1 nélkül a 3DES-t még felismeri. A titkosításnál viszont már bonyolultabb a helyzet: míg az XP hajlandó a DESX algoritmusig lealacsonyodni, a W2K3 legfeljebb a 3DES-ig „butítható". Ennek két módja van; egyrészt az AES használata helyett a W2K3 tartományban előírhatjuk a 3DES használatát *(ez azonban nem csak az EFS-re, hanem az IPSec-re is hatással lesz)*. Ehhez egy házirendben *(különálló gépen érdemes erre a Local Security Policyt, tartományi tagok esetén a Default Domain Policyt használni)* keressük meg és engedélyezzük a következő beállítást: *Local Policies Security Options System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing.*

Ha csak az EFS titkosítóalgoritmusát szeretnénk meghatározni, a registry-be kell belenyúlnunk és létrehoznunk az alábbi értéket:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\
↳ CurrentVersion\EFS\AlgorithmID
```

Értékei pedig az alábbiak lehetnek *(REG_DWORD)*:

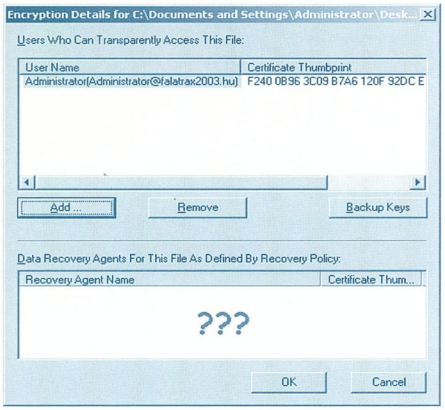
- 0x6610: AES
- 0x6603: 3DES
- 0x6604: DESX *(ez az érték W2K3-ban nem érvényes)*

Alapértelmezett Data Recovery Agent

A Data Recovery Agent *(DRA)* tanúsítványa arra való, hogy a felhasználó által titkosított fájlokat egy arra kijelölt személy *(maga a DRA)* akkor is ki tudja nyitni, ha a felhasználó kulcsa valamilyen oknál fogva nem érhető el. Ennek feltétele, hogy a fájl titkosításakor a titkosító kulcsot a felhasználón kívül a DRA*(-k)* kulcsával is tároljuk. Ez a biztonsági eljárás a Windows 2000-ben nem kerülhető meg: az EFS ott addig nem volt hajlandó titkosítani, míg nem volt legalább egy DRA a rendszerben. Az alapértelmezett DRA az újonnan telepített, nem tartományi tag Windows 2000-ken a helyi, tartományban pedig a tartományi rendszergazda volt; a *(File Recovery típusú)* DRA tanúsítvány a számítógép illetve a tartomány telepítésekor automatikusan létre is jött a rendszergazda profiljában.

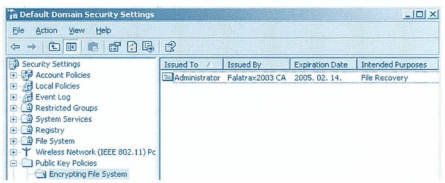
Ez a gyakorlat a Windows XP-ben megszűnt, és már az alapértelmezett DRA tanúsítványokat sem hozza létre az operációs rendszer. Erre azért volt szükség, mert az automatikusan keletkezett, rendszergazdai profilban megbúvó helyreállító tanúsítványok biztonsági problémát jelentettek az ellopott notebookok és egyéb módon megszerzett számítógépek esetén *(Windows 2000-ben első dolgunk legyen a helyreállító tanúsítványt privát kulcsostul flopira exportálni, majd a privát kulcsot törölni a rendszerből!)*. Szükség esetén a lemez a páncélszekrény mélyéről előkaparható, importálható, viszont addig is a fájljainkat valamelyest biztonságban tudhatjuk.

Ne felejtsük el, hogy az XP nemcsak a tanúsítványok létrehozásától tekint el, hanem zokszó nélkül titkosít fájlokat akkor is, ha nincs elérhető DRA a közelben. Ha egy így titkosított fájl tulajdonosának privát kulcsa elveszik, a fájl tartalmára keresztet vethet.



Nem egy életbiztosítás: Windows XP-ben nincs alapértelmezett Recovery Agent!

A helyreállító ügynök tanúsítványát tartományi környezetben a Default Domain Policy biztonsági beállításai között határozhatjuk meg:



Az alapértelmezett helyreállító ügynök tanúsítványa az alapértelmezett tartományi házirendben

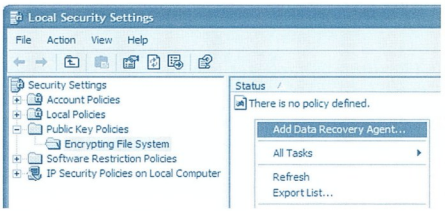
Ehhez persze arra van szükség, hogy a helyreállító ügynök *(magyarán a rendszergazda)* rendelkezzen File Recovery típusú tanúsítvánnyal. Ezt kérheti-kaphatja a vállalati CA-tól *(ha van)*, vagy generálhatja magának. Hasonló a helyzet a nem tartományi tag Windows XP-k esetén: mielőtt DRA-t hoznánk létre, a rendszergazda számára generálnunk kell egy file recovery típusú tanúsítványt. Ehhez a cipher parancsot használhatjuk:

```
C:\>cipher /r:recagent
Type in the password to protect your .PFX file:
Please retype the password to confirm:

Your .CER file was created successfully.
Your .PFX file was created successfully.
C:\>_
```

A parancs generál *(de nem telepít!)* egy File Recovery típusú, ún. „önaláírt" tanúsítványt az aktuális felhasználó nevében-nevére. Ezt a tanúsítványt a privát kulcs nélkül elmenti az általunk megadott nevű, .cer kiterjesztésű fájlba. A privát kulcsot is tartalmazó tanúsítvány pedig az azonos nevű .pfx fájlba kerül. Ezután még két lépés van hátra:

- A tanúsítványt privát kulcsostul telepítenünk kell a felhasználó tanúsítványtárába *(ehhez kattintsunk kettőt a .pfx fájlra és menjünk végig a tanúsítványimportáló varázslón)*
- A publikus kulcsot tartalmazó tanúsítványt pedig be kell állítanunk, mint alapértelmezett DRA tanúsítványt.

Ezt tartományi környezetben az előbb már látott Default Domain Policy-ben, helyi gépen pedig a Local Security Settings megfelelő helyén tehetjük meg.



Helyreállító ügynök „telepítése" különálló Windows XP-ben

Az „Add Data Recovery Agent..." parancs kiválasztása után a varázslóval keressük meg a cipher által generált .cer fájlt, és importáljuk a házirendbe.

Az EFS, a smart card és egyéb biztonsági eszközök

Az EFS csak a beépített Microsoft Base, Strong illetve Enhanced Cryptographic Service Provider *(CSP)* modulokat képes és hajlandó használni. Ez azt is jelenti, hogy az EFS jelen pillanatban nem működik együtt semmilyen külső biztonsági eszközzel, legyen az smart card, biztonsági USB token, vagy bármi más olyan bővítmény, ami saját CSP-t *(„biztonsági eszközmeghajtót")* használ.

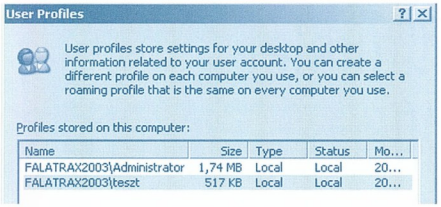
Fájlok titkosítása a kiszolgálókon – Windows megosztáson keresztül

A fájlok titkosításához a felhasználó publikus, míg dekódolásukhoz privát kulcsára van szükség. A Windows megosztott mappákban titkosított fájlokat maga a kiszolgáló titkosítja ki-be *(ez azt is jelenti, hogy a titkosított fájlok a hálózaton „titkosítatlanul" közlekednek)*. Azonban még maga a titkosítás sem egyszerű, hiszen a fájl kezeléséhez a kiszolgálónak szüksége van a felhasználó privát kulcsára. Az pedig a felhasználó profiljában található, ezért a kiszolgálónak a profilt meg kell szereznie. Ennek három módja van:

- a felhasználó vándorló *(roaming)* profilt használ, vagy:
- a felhasználó már belépett az adott kiszolgálóra és létezik a profilja
- a felhasználó még nem lépett be az adott kiszolgálóra, ezért a kiszolgáló létrehoz *(!)* számára egy „üres" profilt.

Ha a profil megvan, és található benne megfelelő tanúsítvány *(és privát kulcs)*, a kiszolgáló azt használja, ha azonban nincs ilyen, akkor *(a W2K3-nak)* megint két lehetősége van:

- Ha van elérhető vállalati tanúsítványkiadó szervezet *(CA)*, online a felhasználó nevében kér egy megfelelő tanúsítványt *(!!!)* és tárolja azt a felhasználó profiljában
- Ha nincs elérhető CA *(vagy a kiszolgáló nem W2K3)*, generál egy önaláírt tanúsítványt, és ugyancsak tárolja a felhasználó profiljában.

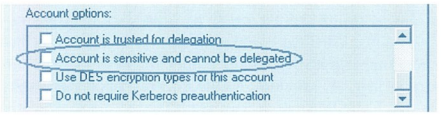


Bár a tesztfelhasználó soha nem „járt" a kiszolgálón, az első titkosítás alkalmával EFS létrehozta az ő kis „miniprofilját"

A titkosítás ezután a „hagyományos" módon megy végbe. A fenti lépések azonban különleges biztonsági beállításokat igényelnek:

- a kiszolgálónak meg kell engednünk, hogy más felhasználó nevében eljárhasson;
- a felhasználó beállításai között nem lehet letiltva a lehetőség, hogy a nevében eljárjanak.

Mindkét opciót természetesen az Active Directoryban találjuk meg, az előbbit az adott számítógép tulajdonságlapjának első oldalán, „Trust Computer for delegation" néven – ezt az EFS „kiszolgálók" számára engedélyeznünk kell *(ez tartományvezérlőknél alapértelmezésben be van kapcsolva)*. A másik beállítást pedig a felhasználó tulajdonságlapjának „Account" oldalán, az Account options listában találjuk.

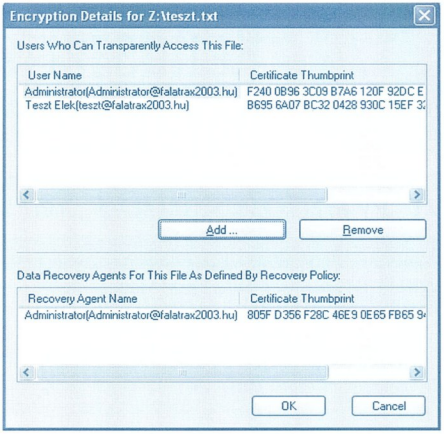


Ha a kijelölt opció a felhasználónál be van állítva, a távoli EFS nem fog működni

Az EFS használatához az „Account is sensitive and cannot be delegated" beállítást ki kell kapcsolni. A felette látható „Account is trusted for delegation" opciót viszont az EFS-hez nem kell bekapcsolni.
Ha ezek a feltételek fennállnak, a távoli EFS titkosításnak már nincs akadálya. Arra figyeljünk azonban, hogy – hacsak a felhasználónk nem vándorló profilt használ – a kiszolgáló által generált kulcspár csak a kiszolgálón található profiljában lesz megtalálható *(tehát szükség esetén onnan kell exportálni is)*.

Egy titkosított fájl, több felhasználó

A Windows XP újdonsága volt, hogy egy titkosított fájlhoz több felhasználót is hozzárendelhettünk. Miután a titkosításhoz mindössze a felhasználó nyilvános kulcsára van szükség *(az pedig megtalálható a közzétett tanúsítványokban)*, ehhez nem kell az „új" felhasználó közvetlen közreműködése. Elég, ha a tanúsítványához és publikus kulcsához hozzáférünk; erre viszont tökéletes hely az Active Directory; ide ugyanis minden, a vállalati CA által kiadott tanúsítvány automatikusan bekerül, és „kézzel" is importálhatunk újabb tanúsítványokat. Az új felhasználók felvétele tehát valóban nem több mint néhány kattintás.



A fájl titkosításához felhasznált tanúsítványok, és a tartalomhoz hozzáférni képes felhasználók listája

Ugyanebben a dialógusablakban végre tisztán látható, hogy pontosan kinek és pontosan melyik tanúsítványát használta a rendszer a titkosításhoz. A tanúsítvány az „ujjlenyomat" *(Certificate Thumbprint)* segítségével könnyen azonosítható.

Fájltitkosítás webes megosztáson keresztül

Az „új" EFS még egy újdonságot tartalmaz, mégpedig a WebDAV megosztáson keresztüli titkosítást. Ennek több előnye is van a hagyományos fájlmegosztáson keresztüli titkosítással szemben:

- A titkosítást nem a kiszolgáló, hanem minden esetben az ügyfél számítógépe végzi, így megszűnik a profil és kulcsmenedzsment problémája
- A fentiek értelmében a titkosított fájlok tartalma már a hálózaton is titkosítva „utazik" *(!)*
- A WebDAV kommunikáció nem más, mint kibővített HTTP, így sokkal „tűzfalbarátabb", mint a „hagyományos" Windows hálózati forgalom

A módszert azonban lapzártáig nem sikerült működésre bírni *(ez a béta szoftverek egyik érdekes sajátossága)*, ezért erre a témára egy későbbi számunkban *(például az új IIS-t bemutató cikkek során)* még visszatérünk.

Fülöp Miklós
mick@netacademia.net

Kapcsolódó tanfolyamaink:
Windows 2003 Server Expert Workshop

Rendszerfelügyelet parancssorból

A Windows 2003 Server WMI szolgáltatásai

A WMI *(Windows Management Instrumentation)* rendszerfelügyeleti felületet 1996-ban, akkor még WBEM *(Web Bases Enterprise Management)* néven kezdte el kidolgozni a Microsoft, a Compaq, a BMC, a Cisco és az Intel. A cél egy univerzális, különféle hardver- és szoftverelemekhez egyaránt használható felület létrehozása volt. A Microsoft most még rátett egy lapáttal: a Windows 2003 Serverben egy nem is akármilyen parancssori eszközt is kaptunk hozzá.

A WMI történelme a Windowsban

A Windows Management Instrumentation névre átkeresztelt technológia a Windows NT 4.0 Service Pack 4 CD-n jelent meg „élesben" először – leginkább csak, mint programozói felület. Ezt használta ki többek között a később megjelent Systems Management Server 2.0 is. A Windows 2000, majd az XP megjelenésével a WMI tovább bővült, fejlődött, de a lehetőségek igazán csak a programozó kedvű rendszergazdák szívét dobogtatták meg. A WMI, mint rendszerszolgáltatás a Windows 2000-től kezdődően minden Windowsnak *(a Milleniumnak is)* része, a jelenlegi legújabb WMI-támogatás Windows 95-98-NT4-hez pedig letölthető a **[wmicore]** címről.

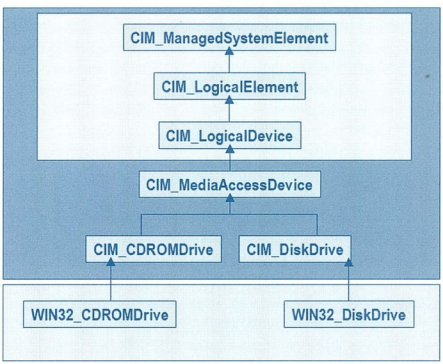
És hogy mindez mire jó? Nos, rendszergazdai jogok birtokában *(merthogy ez mégiscsak egy rendszerfelügyleti eszköz)*, akár távoli gépről is, a legapróbb részletekig lekérdezhetjük a hardver- és szoftverkörnyezet tulajdonságait, sőt sok esetben módosíthatjuk is azokat.

```
Command Prompt
Microsoft Windows [Version 5.2.3757]
(C) Copyright 1985-2003 Microsoft Corp.

C:\>wmic /node:w2003xp desktopmonitor get screenwidth,
    screenheight
ScreenHeight  ScreenWidth
600           800

C:\>wmic /node:w2003xp quotasetting where
    "VolumePath='C:\\'" set State=1, DefaultLimit=1048560
Updating property(s) of '\\W2003XP\ROOT\CIMV2:Win32_QuotaSet
ting.VolumePath="C:\\"'
Property(s) update successful.

C:\>wmic /node:w2003xp /privileges:ENABLE os WHERE
    Primary='True' call Win32Shutdown 5
Executing (\\W2003XP\ROOT\CIMV2:Win32_OperatingSystem.Name="
Microsoft Windows XP Professional|C:\WINDOWS|\\Device\\Hard
disk0\\Partition1")->Win32Shutdown()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ReturnValue = 0;
};
```

Kedvcsinálónak: távoli Windows XP munkaállomás képernyőfelbontásának lekérdezése, a C:\ meghajtón alkalmazott kvóta bekapcsolása, illetve a Windows leállítása – egy-egy sorban

A WMI felépítése, működése

Ahhoz azonban, hogy a fenti parancsokat csuklóból adjuk ki, meg kell ismernünk a WMI belső felépítését. A WMI nagyon univerzális: ugyanolyan módon kérdezhetjük le a router hűtőventillátorának fordulatszámát *(persze csak ha a router ezt lesz kedves elárulni nekünk)*, mint ahogyan a Windowsra telepített hotfixek listáját. Az univerzális működés feltétele viszont egy séma, aminek segítségével „megfoghatjuk" a kezelni kívánt eszközöket – legyen az hardveres, vagy szoftveres elem. Ehhez a WBEM-szabvány kialakításakor a készítők elkészítettek egy információs modellt, sémát, ez a CIM *(Common Information Model)*. A séma rengeteg eszköz definícióját tartalmazza, mégpedig osztályhierarchia formájában *(azaz nem úszzuk meg itt sem az objektumorientált felfogást, az osztály, egyed, öröklés fogalmát)*. Ez a modell egyébként tovább is bővíthető, és ezt a Microsoft meg is tette, természetesen az eredeti CIM-re alapozva – a Microsoft által elkészített bővítményben természetesen leginkább a Windows-specifikus objektumosztályokat találjuk meg. Hogy kicsit világosabb legyen, miről beszélünk, lássunk egy példát. A Windows CD-ROM meghajtóját jelképező Win32_CDROMDrive osztály családfája az alábbiak szerint vezethető le:



A Win32_CDROMDrive családfája

A Win32_CDROMDrive osztály tehát közvetlenül a CIM_CDROMDrive leszármazottja, de a családfáját visszavezethetjük például a CIM_LogicalDevice osztályig is. Hogy mi értelme van ennek az öröklésnek? Minden osztálytípus saját jellemzőket definiálhat, de örökli a szülőktől származó jellemzőket is. A Win32_CDROMDrive például rendelkezik egy

„Caption" jellemzővel *(ami a meghajtó szöveges azonosítója)* – ezt még a CIM_ManagedSystemElement „ősétől" örökölte; de ugyanígy van például „VolumeName" *(kötetnév)* jellemzője is, ami pedig a Win32_CDROMDrive osztály sajátja. Azért ne ijedjünk meg túlságosan: az esetek nagyon nagy százalékában csak a „kész", általában Win32_... objektumokat használjuk fel, ráadásul a wmic.exe még tovább egyszerűsíti majd a dolgunkat.

Osztályok és egyedek

Az eddigiekben a különféle objektumok modellezéséhez használt osztályhierarchiáról volt szó. Az osztályok *(class)* azonban önmagukban nem használhatók a valódi objektumok kezeléséhez. A menedzselt objektumot valójában jelképező fogalom az „egyed" *(instance)*. Természetesen minden egyed valamelyik osztályba tartozik, és jelképez egy-egy, a valóságban is létező objektumpéldányt. Tegyük fel például, hogy a rendszerünk két CD-ROM-meghajtót tartalmaz. A sémában Win32_CDROMDrive **osztály** természetesen ekkor is csak egy van, viszont a WMI adatbázisában ekkor **két** Win32_CDROMDrive osztályú **egyed** található. Ha csak az egyik CD-ROM-meghajtó jellemzőire vagyunk kíváncsiak, ki kell választanunk azt az egyedet, amely az adott objektumot jelképezi.

Hogyan különböztetjük meg egymástól az osztály- és egyedobjektumokat? Ha bárhol egyszerűen az osztály nevére hivatkozunk, természetesen az osztályobjektumról beszélünk. Az egyedek megkülönböztetéséhez viszont meg kell értenünk még egy fogalmat. Minden osztály tartalmaz egy, úgynevezett kulcs *(key)* jellemzőt, aminek értéke minden egyes egyed esetében *(nomen est omen)* **egyedi**. Hogy az adott osztálynak melyik jellemzője a kulcs, az osztály definíciója dönti el. A Win32_CDROMDrive osztály esetén ez például a „DeviceID". A kulcsjellemző segítségével mindig pontosan nyakoncsíphetjük a kívánt objektumot.

Írható jellemzők és metódusok

Az objektumok jellemzői közül a legtöbb csak olvasható típusú *(gondoljunk csak bele, egy memóriamodul méretét például nincs sok értelme módosítani)*, de akad néhány írható is. Az objektumok a jellemzők mellett eljárásokkal, **metódusokkal** is rendelkeznek; ezeket az eljárásokat a WMI alrendszer értelmezi és hajtja végre. A rendszerszolgáltatásokat jelképező Win32_Service osztályú egyedek például rendelkeznek StartService és StopService metódusokkal; ezek meghívása elindítja, illetve leállítja az adott szolgáltatást. A metódusok többségét az egyedeken hívjuk meg *(hiszen például egy konkrét szolgáltatást szeretnénk leállítani)*, de előfordulhat az is, hogy egy osztályobjektum metódusát hívjuk meg *(ilyen például a Win32_Process osztály Create metódusa)*.

A WQL lekérdezőnyelv

A WQL *(WMI Query Language)* nyelvet arra fejlesztették ki, hogy segítségével egyértelműen hivatkozhassunk egy adott osztály adott egyedére. A nyelv viszonylag egyszerű, és hasonlít az SQL-re. Az egyszerűbb lekérdező WQL-utasítások szintaxisa így alakul:

```
SELECT * FROM <osztály> [WHERE
<jellemző><operátor><érték> [<logikai művelet>
<jellemző><művelet><érték> […]]]
```

, ahol:

- <osztály>: a WMI osztály neve
- <jellemző>: a fenti WMI osztály egy jellemzőjének neve
- <operátor>: =, <>, <, >, <=, >=, IS, IS NOT *(utóbbi kettő csak a Null értékkel)*
- <logikai művelet>: AND, OR, NOT, stb.

Néhány példa:

- A rendszerben található összes Win32_CDROMDrive osztályú egyed:

```
SELECT * FROM Win32_CDROMDrive
```

- Az „Administrator" felhasználói fiókja:

```
SELECT * FROM WIN32_UserAccount WHERE
Name="Administrator"
```

- Az összes éppen futó processz, amelynek prioritása nagyobb, mint 8:

```
SELECT * FROM Win32_Process WHERE Priority > 8
```

A WQL-ből kezdésnek ennyi elég is *(a további finomságokra később még visszatérünk)*.

Ismerkedés a wmic.exe paranccsal

A Windows 2003 Server wmic.exe parancsa megkönnyíti a WMI használatát. Maga a parancs az egészen egyszerű „egysorosoktól" ez egészen összetett, formázott XML kimenetű adatgyűjtésig nagyon sok mindent tud. Kezdjük a legelején! A parancsnak két üzemmódja van: az interaktív és a non-interaktív mód. A működése persze mindkét módban ugyanaz, a különbség mindössze annyi, hogy interaktív módban belépünk a WMIC „promptba", majd ott is maradunk addig, amíg ki nem adjuk az exit vagy quit parancsot:

```
C:\>wmic
wmic:root\cli>os get caption
Caption
Microsoft(R) Windows(R) Server 2003, Standard
↳ Edition
wmic:root\cli>quit
C:\>_
```

A másik, non-interaktív módban a parancsot a wmic.exe paramétereként adjuk át. A WMIC ilyenkor végrehajtja az utasítást, majd rögtön ki is lép:

```
C:\>wmic os get windowsdirectory
WindowsDirectory
C:\WINDOWS
C:\>_
```

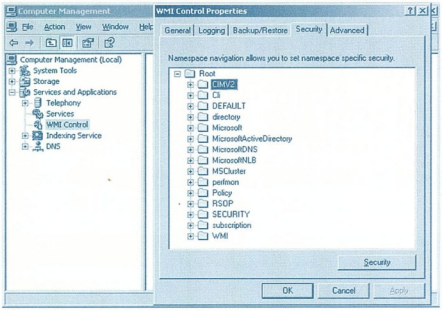
A továbbiakban – hacsak külön nem jelezzük – a non-interaktív módot használjuk.

Szükséges jogosultságok a WMIC helyi futtatásához

A WMIC parancsot rendszergazdai jogokkal futtathatjuk. Ha a felhasználó nem rendszergazda, az alábbi jogosultságokkal kell rendelkeznie:

- Teljes írásjog a HKLM\Software\Microsoft\WBEM illetve a HKLM\Software\Microsoft\WBEM\WMIC registry kulcsokra
- Teljes írásjog a root/cli illetve root/cimv2 névterekre.

A WMI névterekre a folytatásban még visszatérünk. A névterek hozzáférési jogait a Computer Management eszköz Services and Applications WMI Control Properties Security oldalon állíthatjuk be. Itt válasszuk ki a Root\CIMv2 illetve a Root\Cli névtereket és kattintsunk a Security gombra.



Hozzáférési jogosultságok a WMI névterekhez – a rendszergazdának „gyárilag" megvan

WMIC aliasok

A szemfüles olvasónak már nyilván feltűnt, hogy a fenti példákban az operációs rendszer néhány jellemzőjét kérdeztük le. Igen ám, de milyen osztály jellemzőiről van szó? Hol van itt a CIM séma? Mi az az „os" osztály? Nos, az „os" osztály valóban nem létezik, a lekérdezések a háttérben a Win32_OperatingSystem egyetlen létező egyedének jellemzőire vonatkoztak. Az „os" egy **alias**, amivel a WMIC igyekszik megkönnyíteni a rendszergazda munkáját. Alapértelmezésben mintegy 80 ilyen alias létezik, az aliasok listáját és „magyarázatát" a wmic.exe /? parancs elárulja nekünk. Ha még konkrétabban szeretnénk tudni, mi történik az aliasok feloldásakor, írjuk be:

```
C:\>wmic alias os get target
Target
Select * from Win32_OperatingSystem
```

A fenti parancs lekérdezi az „os" nevű alias-objektum *(mert ez is egy WMI-egyed!)* target paraméterét, és lám, máris egy WQL lekérdezésbe futottunk! Innentől már világos a kép: amikor a WMIC-ben az „os" aliast használjuk, valójában a Win32_OperatingSystem osztály egyedeit kezeljük *(még szerencse hogy ebből csak egy van)*. Ez a lekérdezés *(amiben az „os" helyére tetszőleges másik aliast is behelyettesíthetünk, még magát az „alias"-t is :-))* elárulja nekünk, hogy igazából melyik osztály egyedeivel van dolgunk, és hogy azt hol keressük a WMI osztályok leírása között, a WMI referenciában *(a referencia web-címe: **[wmiref]**)*.

Objektumok jellemzőinek lekérdezése: a get művelet

A WMIC utasításban először az objektum*(ok)*ra utaló aliast, majd a velük végezni kívánt műveletet kell megadnunk. A „get" művelet az objektumok jellemzőinek lekérdezésére szolgál. A művelet után meg kell adnunk a lekérdezni kívánt jellemzők nevét *(több jellemző esetén vesszővel elválasztva)*:

```
C:\>wmic os get version
Version
5.2.3763
```

A WMIC az adatokat „belül" XML formátumban kezeli. Az, hogy ennek eredményét mi milyen formában látjuk, az adott alias alapértelmezett beállításaitól függ. Az eredmény formátumát mi is meghatározhatjuk a get művelet /format kapcsolója segítségével. A /format után meg kell adnunk a kívánt formátum nevét *(vagy a formátum elkészítéséhez használt .xsl fájl nevét)*, ami alapértelmezésben a következők valamelyike lehet *(súgó: wmic os get /format /?)*:

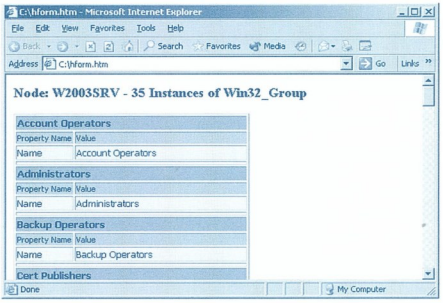
Név	Leírás
csv	Vesszővel elválasztott értékek, soronként egy egyed
hform	HTML táblázat, minden érték új sorban
htable	HTML táblázat, minden sor egy egyed, minden érték egy oszlop
hmof	MOF formátum *(HTML-ben)*
hxml	XML formátum *(HTML-ben)*
list	Szöveges kimenet, soronként egy jellemző neve és értéke
table	Szöveges, soronként egy egyed, oszloponként egy jellemző értéke
value	Mint a list
rawxml	Valódi XML, a lekérdezés eredményének eredeti formája *(irányítsuk .xml kiterjesztésű fájlba, így az IE megnyitja nekünk)*. A fájl tartalma egyébként nagyon sokat elárul a WMIC működéséről.

A WMIC kimenetének formátumai (/format: kapcsoló)

Próbáljuk ki a különböző formátumokat például az alábbi parancs segítségével:

```
wmic group get name, SID /format:<formátumnév>
```

Ha nem adunk meg formátumot, akkor az adott alias, illetve az adott lekérdezés típusának megfelelő formátumban kapjuk az eredményt *(ez általában a list vagy a table)*.



A csoportok listája HTML formátumú kimenetben

Ilyen kimenet-formázió XSL fájlt egyébként mi magunk is előállíthatunk *(lásd később)*.

A get művelet további kapcsolói

A /format mellett a get és a list művelet további kapcsolókat is támogat, többek között:

- /value: az egyed összes jellemzőjének értéke, a kimenet „value" formátumú
- /all: az egyed összes jellemzőjének értéke, a kimenet „table" formátumú
- /every:<mp>: a lekérdezés megismétlése „mp" másodpercenként
- /?[:full]: az osztály jellemzőinek listázása (a :full megadása esetén magyarázattal együtt, például:

```
wmic os get /?
wmic os get /?:full
```

Objektumok jellemzőinek listázása: a list művelet

Az objektumok jellemzőit a "list" művelet segítségével is lekérdezhetjük. A különbség annyi, hogy míg a get műveletnél mi magunk határozhatjuk meg a lekérdezendő jellemzőket, a list parancsban "csomagok" vannak, ezek pedig az alábbiak *(azt, hogy melyik alias esetén melyik csomagban pontosan melyik jellemzők vannak, a wmic <alias> list /? paranccsal tudjuk meg)*:

- brief: az osztály legfontosabb jellemzői
- full: az osztály összes jellemzője *(ez ugyanaz, mintha a get /all utasítást adtuk volna ki)*
- instance: az egyedi jellemzők listázása
- status: státuszjellemzők, állapotjellemzők
- system: rendszerjellemzők
- writeable: az írható jellemzők
- egyéb: az alias egyedi jellemzőcsomagjai *(lásd az adott alias helpjét)*

A list művelet a get-hez hasonlóan támogatja az /every és /format kapcsolókat is. Néhány példa:

```
wmic bios list full
wmic volume list brief
wmic cpu list status /format:value
wmic os list free /format:value
```

Egy adott egyed kezelése *(WHERE paraméter)*

Láthattuk, hogy az aliasok a rendszerban található, adott osztályú egyedek sokaságát adják vissza *(SELECT * FROM <osztály>)*. A lekérdező jellegű műveletek esetén ezzel általában nincs is különösebb baj, legfeljebb a válasz kicsit bőségesebb lesz a vártnál és mazsolázunk kell. Vannak helyzetek azonban, amikor ez nem engedhető meg, és egészen pontosan ki kell választanunk a kívánt egyedet. Ehhez a WMIC WHERE paraméterét használhatjuk, amit közvetlenül az alias neve után kell megadnunk, például:

```
C:\>wmic service WHERE "Name='Alerter'" get state
State
Stopped
```

Ami nem is túl meglepő, ha visszagondolunk arra, hogy egy alias valójában nem más, mint egy WQL lekérdezés „eleje". A WHERE paraméterben persze sokkal bonyolultabb és összetettebb feltételeket is megadhatunk, csak két dologra vigyázzunk:

- Ha a feltétel speciális karaktert *(-, /, stb.)* tartalmaz, az egész WHERE után következő feltétellistát tegyük idézőjelek közé *(ez jó ha szokásunkká válik, mert sok bosszúságtól kímélhet meg a későbbiekben!)*
- A backslash *(\)* karaktert mindig duplázva írjuk

```
C:\> wmic datafile WHERE "Hidden=TRUE AND
↳ Path='\\windows\\' AND FileSize>10000" get
↳ Name, FileSize
FileSize  Name
49104     c:\windows\lanma256.bmp
49104     c:\windows\lanmannt.bmp
```

A kimenet átirányítása *(WMIC kapcsolók)*

A parancs kimenetét fájlba irányíthatjuk a klasszikus módon *(>)*, vagy használhatjuk a wmic parancs /output illetve /append kapcsolóját *(előbbi felülírja, utóbbi hozzáfűzi a kimenetet)*, a következő értékekkel: stdout *(képernyő, alapértelmezés)*, clipboard *(a Windows vágólap)*, vagy fájlnév, például:

```
wmic /output:clipboard os get caption
wmic /append:names.xml os get name /format:rawxml
```

Figyeljük meg, hogy a WMIC kapcsolóját a wmic utasítás, a get művelet kapcsolóit pedig a művelet paraméterei után adtuk meg.

Jellemzők beállítása: a set művelet

Lekérdezni már tudunk, most következzen a beállítások módosítása. A WMI objektumok egy része nem túl sok írható jellemzőt tartalmaz, de ezeket lekérdezhetjük például a

```
wmic <alias> set /?[:full]
```

parancs segítségével. A bootmenü várakozási idejének átállítása például így fest:

```
C:\>wmic computersystem set SystemStartupDelay=10
Updating property(s) of
'\\W2003SRV\ROOT\CIMV2:Win32_ComputerSystem.Name=
"W2003SRV"'
Property(s) update successful.
```

A set művelet után az átállítani kívánt jellemző nevét és értékét egyenlőségjellel kell megadnunk. Több jellemző módosítása esetén az „egyenletek" közé vesszőt kell tenni. Ha egy adott aliasból *(osztályból)* több, mint egy egyed létezik, persze használnunk kell a WHERE paramétert is:

```
C:\>wmic quotasetting WHERE "VolumePath='C:\\'"
↳ set State=2, DefaultLimit=10485760
Updating property(s) of
↳ '\\W2003SRV\ROOT\CIMV2:Win32_QuotaSetting.
↳ VolumePath="C:\\"'
Property(s) update successful.
```

A fenti parancs a C:\ meghajtón bekapcsolja a kvótarendszert *(mégpedig úgy, hogy meg is akadályozza az írást azok számára, akik átlépik az engedélyezett értéket)*; az alapértelmezett kvótát pedig 10MB-ra állítja be. A State jellemző értéke a referenciával ellentétben nem szöveges, hanem számszerű, de indirekt módon könnyen kikísérletezhető, hogy melyik érték mit jelent *(hiszen lekérdezni ugye már tudunk?)*.

Metódusok hívása: a call művelet

A WMIC „call" műveletének segítségével az objektumok metódusait hívhatjuk meg *(persze amennyiben az adott objektumnak létezik metódusa)*. A call műveletnek nincsenek speciális paraméterei, a call után egyszerűen csak át kell adnunk a meghívni kívánt metódus nevét, majd vesszővel elválasztva az esetleges paramétereket. Az egyes aliasok *(osztályok)* esetében hívható metódusok listáját természetesen a

```
wmic <alias> call /?
wmic <alias> call /?:full
```

parancsok valamelyikével csalhatjuk elő. A metódusok meghívása azonban már nem ennyire értelemszerű. A call művelet ugyanis egyedekre értelmezendő, ezért ha elfelejtjük a WHERE használatát, a parancs hibát fog okozni *(az nem baj, ha a WHERE kitétel több egyedet ad vissza, ilyenkor a metódus minden visszaadott egyedre meghívódik)*. A másik dolog, ami nehezíti a metódusok hívását, hogy a /? visszaadja ugyan a metódus paramétereinek listáját és azok típusait, de lehetséges értékeit nem! Ezért legkésőbb a metódusok hívásánál elkerülhetetlen, hogy fellapozzuk a WMI referenciát **[wmiref]** és megkeressük benne a metódus pontos leírását. A

```
wmic process call Create "calc.exe"
```

paranccsal indítsunk például néhány Számológépet, majd a

```
wmic process WHERE Name='calc.exe' call Terminate
```

paranccsal szépen, egy lépésben ki is lőhetjük őket. A process alias *(Win32_Process osztály egyedei)* Terminate metódusa paraméter nélkül használható, és a Create metódusnak csak egy kötelező paramétere van, mégpedig az indítani kívánt processz neve. Figyeljük meg, hogy a Create hívásakor nem használtuk a WHERE paramétert, mert a Create metódus nem kötődik szigorúan az egyedekhez. A processz leállításához *(Terminate)* viszont már meg kell mondanunk, hogy pontosan melyik egyedekre gondoltunk. Példa egy paraméterezett metódushívásra:

```
wmic nteventlog WHERE "LogfileName='Application'"
↳ call BackupEventlog "c:\applog.evt"
```

Ha több átadandó paraméterünk van, azokat vesszővel elválasztva kell felsorolnunk a metódus neve után.

Műveletvégzés távoli gépen *(WMIC kapcsolók)*

Itt az ideje, hogy kimozduljunk saját számítógépünk hatósugarából. A WMIC távoli számítógépek esetén is működik, amennyiben a WMIC-t futtató „kliens" és a felügyelt „szerver" gép között Windows hálózati kapcsolat *(DCOM, RPC)* építhető ki. Jogosultság szempontjából ugyanaz igaz, mint a helyi felhasználásnál: helyi rendszergazdai jogokkal kell rendelkeznünk. Ha Windows tartományban vagyunk, és a WMIC-t tartományi rendszergazdaként futtatjuk, nem fog meglepetés érni. Fontos, hogy a WMIC-nek nem kell a célgépen telepítve lennie, elég, ha ott a WMI szolgáltatás fut.
A távoli menedzsment három legfontosabb kapcsolója a /node, a /user és a /password:

```
C:\>wmic /node:w2003xp /user:teszt /password:titok
↳ os get caption
Caption
Microsoft Windows XP Professional
```

A /user és a /password kapcsolók önmagukért beszélnek. A /node: kapcsoló után, vesszővel elválasztva kell megadnunk a kezelni kívánt számítógép*(ek)* nevét vagy IP címét. A listába felvehetünk szöveges fájlokat is, ekkor a fájl neve elé @-ot kell írnunk. A fájl természetesen számítógépneveket tartalmazzon, soronként és/vagy soron belül vesszővel elválasztva egyet-egyet. Kiragadott példa a súgóból:

```
/node: "computer1","computer2",@"c:\names.txt"
```

Ennyit mára az elméletből, következő számunkban innen folytatjuk. Lazításképpen bemutatunk egy hasznos metódust, illetve annak részletes használatát:

Felhasználó kijelentkeztetése, számítógép leállítása vagy újraindítása

A fentieket mind-mind megtehetjük a Win32_OperatingSystem osztály Win32Shutdown metódusával. A cikk elején is látott példa a W2003XP nevű számítógépet állítja le, mégpedig könyörtelenül:

```
C:\>wmic /node:w2003xp os WHERE Primary=True call
Win32Shutdown 5
```

A Win32Shutdown metódusnak egy értékes paramétere van, ami a követendő eljárást jelzi:

Érték	Leírás
0	Kijelentkezés *(Log Off)*
0+4 = 4	Kijelentkezés rákérdezés nélkül *(Forced Log Off)*
1	Leállítás *(Shutdown)*
1+4 = 5	Leállítás rákérdezés nélkül *(Forced Shutdown)*
2	Újraindítás *(Reboot)*
2+4 = 6	Újraindítás rákérdezés nélkül *(Forced Reboot)*
8	Kikapcsolás *(Power Off)*
8+4 = 12	Kikapcsolás rákérdezés nélkül *(Forced Power Off)*

A rákérdezés nélküli műveletek esetén a Windows nem várja meg, hogy a felhasználó elmentse a munkáit, hanem azonnal végrehajtja a műveletet.

Végül egy jótanács a folytatásig: az alábbi parancsot:

```
wmic process WHERE "Name IS NOT Null" call
Terminate
```

csak akkor próbáljuk ki, ha előtte mentettünk ;-)!

Folytatjuk...

Fülöp Miklós
mick@netacademia.net

Kapcsolódó tanfolyamaink:
Windows 2003 Server Expert Workshop

A cikkben szereplő URL-ek a **http://technet.netacademia.net/go?** kulcsszó címen érhetők el.

A Microsoft Exchange 2000 útválasztása

Az üzenetek útja egy szervezeten belül és kívül

Egy szervezetben már két Exchange kiszolgáló esetén is felmerül a kérdés: merre mennek az elküldött üzenetek? Gyakran nem is tudja a rendszergazda, mi történik tulajdonképpen több kiszolgáló esetén. A cikkben szó lesz az üzenetek útjáról, az útválasztásról, és az összefüggésekről.

Az alapok

Ahhoz, hogy Exchange 2000 környezetben megértsük az útválasztást, meg kell ismerkednünk néhány alapvető dologgal, például az útválasztócsoportokkal, a csatolókkal és az állapotjelző táblázattal. Menjünk szépen sorban, a cikk a végére fog igazán kikerekedni.

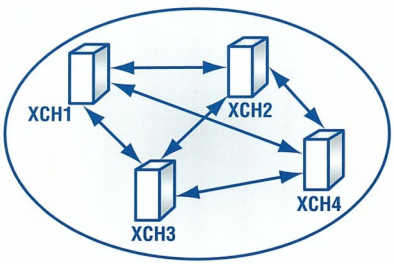
Az Exchange 2000 telepítésekor automatikusan létrejön egy alapértelmezett útválasztócsoport az Adminisztratív csoporton belül és – nem újdonság – **First Routing Group** lesz a neve. Ebbe kerül bele rögtön az első kiszolgáló, és ha nem hozunk létre több útválasztási csoportot *(mert nem kötelező ám)*, minden további kiszolgáló is ebbe a csoportba fog kerülni.

Mikor érdemes több útválasztócsoportot *(Routing Group)* létrehozni?

Amikor valóban több telephelyen szétszórva vannak a kiszolgálóink, köztük a kapcsolat nem állandó *(nem – úgymond – LAN szintű, nem megbízható és/vagy nem gyors)*. Persze más érvek is vezérelhetnek bennünket, például a levélforgalom szabályozása a kiszolgálók közt. Másként megy a kiszolgálók közti üzenetforgalom, ha azok egy, és másként, ha két különböző csoportban vannak.

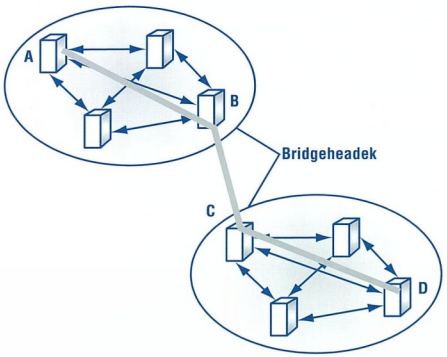
Egy útválasztócsoporton belül a kiszolgálók közvetlenül beszélgetnek egymással, mindenki mindenkivel. Ezt a forgalmat nem lehet időzíteni, azonnali a végrehajtás.

A képen is ez látszik, minden kiszolgáló az összes többivel közvetlen kapcsolatban áll:



Üzenettovábbítás útválasztócsoporton belül

Ha viszont két különböző útválasztócsoportban levő kiszolgáló akar beszélgetni, bizonyos szabályokat be kell tartaniuk, és valószínű, hogy közvetlenül nem is tudnak kommunikálni, hanem az úgynevezett **Bridgehead** kiszolgálókon keresztül tehetik azt meg. A Bridgehead kiszolgálókat csatolók kötik egymáshoz. Jól mutatja ezt a következő ábra is:



Egy üzenet útja

Tegyük fel, hogy Józer Jolán *(akinek a postaládája az **A** jelű kiszolgálón található)*, levelet küld Teszt Eleknek *(akinek pedig a **D** jelű kiszolgálón vannak a levelei)*. A következő lépések történnek:

1. Az **A** jelű kiszolgáló észleli, hogy bizony másik útválasztócsoportba szól a levél, ezért megkeresi a megfelelő bridgehead kiszolgálót és passzolja neki a levelet. Ez a példában a **B** kiszolgáló.
2. A **B** kiszolgáló – bár át tudja küldeni a másik útválasztócsoportba a levelet – nem közvetlenül **D**-vel tárgyal, hanem a **C** kiszolgálónak küldi tovább az üzenetet, mert vele áll közvetlen kapcsolatban.
3. A **C** jelű kiszolgáló fogadja a levelet, és közvetlenül **D**-nek fogja elküldeni, hisz ők már egy útválasztócsoportba tartoznak. Végül Teszt Elek postaládájában landol a levél.

Olyan környezetben, ahol Exchange 5.5 és Exchange 2000 él együtt, az útválasztócsoportoknak kicsit más az értelmezése.

Egyrészt egy csoportban nem lehet több Adminisztratív csoportból kiszolgáló, másrészt az Exchange 5.5 az Exchange 2000 Administrative Groupokat siteoknak képzeli – az útválasztócsoportok számára nem is léteznek.

Exchange natív módban, vagyis amikor már csak Exchange 2000 kiszolgálók vannak a szervezetben, egy útválasztócsoportban több adminisztratív csoportból származó kiszolgáló is szerepelhet.

Érdemes tisztázni, mikor tartozhatnak a kiszolgálók azonos csoportba:

1. Ha a kiszolgálók megbízható, állandó, közvetlen hálózati kapcsolatban állnak egymással *(ez tipikusan LAN-t jelent)* és
2. ugyanabban az Active Direcory erdőben vannak, valamint
3. közvetlen SMTP kapcsolatot tudnak teremteni egymással

Másik útválasztócsoportban levő kiszolgálóval Bridgehead és Routing Group Master kiszolgálókon keresztül kommunikálnak.

Kommunikáció az útválasztócsoportok közt

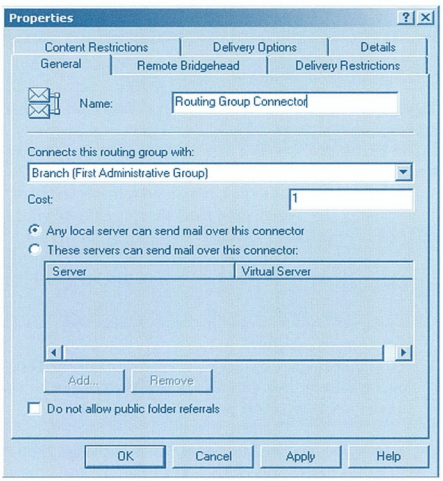
Az útválasztócsoportok közt a kommunikáció nem automatikus. Amikor létrehozunk több útválasztócsoportot, gondoskodni kell a köztük levő kommunikációról is. Alapvetően az üzenettovábbítás a Bridgehead kiszolgálók közbeiktatásával, csatolókon *(Connector)* keresztül történik. A csatoló olyan, mint egy „kábel" két szerver között, amelyen SMTP adatforgalom zajlik. Háromféle csatolót tehetünk két útválasztócsoport közé:

- Routing Group-csatoló
- SMTP-csatoló
- X400-csatoló

A csatolók *(„kábelek")* végein található kiszolgálókat Bridgeheadeknek *(hídfő)* nevezzük. Egy kiszolgáló *(virtuális SMTP szerver)* többszörös Bridgehead is lehet, és egyszerre több szerver *(akár mindegyik)* is lehet Bridgehead egy útválasztócsoportban.

Routing Group Connector

Az útválasztócsoportok összekötésére legegyszerűbben a Routing Group-csatoló használható *(Routing Group Connector)*.



Kézenfekvő megoldás mert:

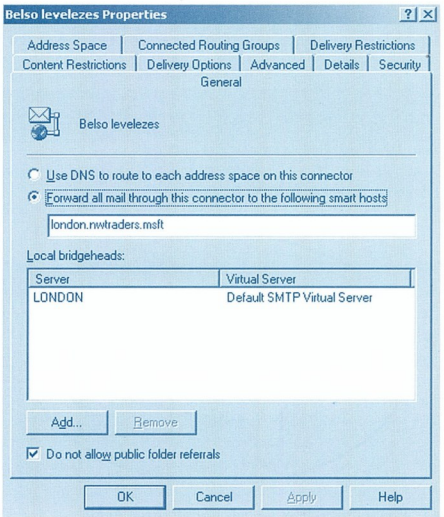
- Több Bridgehead kiszolgáló lehet egy csatoló mindkét végén, ezért több Exchange kiszolgáló esetén hibatűrést illetve terheléselosztást is tudunk biztosítani.
- A csatoló által használt kommunikációs protokoll alapvetően az SMTP (például, ha a csatoló mindkét végén Exchange 2000 van – vagy egy Exchange 5.5 kiszolgáló IMC-vel).
- Képes RPC-vel is kommunikálni, ha egy Exchange 2000-et Exchange 5.5-tel szeretnénk összekötni. Ebben az esetben egyébként Exchange 5.5 oldalról Site Connector szükséges a működéshez.
- A csatoló működése időzíthető, így jól használható, ha például nagyméretű állományok továbbítását éjszakára szeretnénk időzíteni.
- A csatolón szintén korlátozhatjuk, ha csak rendszerüzenetek *(pl. nyilvános mappák replikációja)* vagy egyéb üzenetek továbbítását szeretnénk biztosítani.

Hátránya, hogy biztonsági beállításokat nem tesz lehetővé. *(Ha például SSL-lel szeretnénk levelezni, azt a virtuális kiszolgáló szintjén kell beállítani.)* Ugyanakkor megjegyzem, hogy a levelek formátuma nem sima szöveg, hanem TNEF *(Transport Neutral Encapsulation Format)*, így avatatlanok nem olyan könnyen férnek hozzá az elkapott csomagok tartalmához.

SMTP Connector

Főként akkor érdemes használni, ha

- Exchange 5.5-ös IMC-vel felszerelt másik útválasztócsoportban levő kiszolgálót szeretnénk a csatoló másik végén látni.
- ETRN vagy TURN segítségével szedjük a leveleket az egyik telephelyen a másikról *(elképzelhető, hogy egy vidéki telephely nem állandó, hanem ISDN kapcsolattal van összekötve a központi telephellyel, és a leveleket a vidéki kiszolgáló időnként ETRN-nel szedi le.)*
- a csatoló szintjén SSL-t szeretnénk használni. Ekkor **csak** az SMTP-csatolót használhatjuk



A Routing Group-csatolóval szemben itt csak a helyi Bridgehead kiszolgálókat tudjuk megadni. Érthető is, hiszen alapvetően MX rekordok alapján tájékozódik a csatoló.
Ezen kívül lehetőség van smarthost beállítására, amikor a kiszolgáló minden levelet egy másiknak továbbít. Amikor két útválasztócsoport közt SMTP-csatolót hozunk létre, figyelmeztet is a System Manager, hogy a másik útválasztócsoportban levő kiszolgálót kell megadnunk smarthostnak a két útválasztócsoport közti kommunikáció biztosításához.

A beállítás bonyolultsága miatt csak akkor érdemes SMTP-csatolót használni, ha valóban olyan funkcióra van szükségünk, amit a Routing Group Connector nem tud biztosítani.

X.400-csatoló

A jó öreg X.400-as csatoló hasznos lehet, ha olyan telephelyeket szeretnénk összekötni, amelyek közt igen kicsi az átviteli sebesség, de azért megbízható. Az X.400-csatoló jól működik nagyméretű levelek esetén is, kicsi átviteli sebesség esetén. Ugyanakkor beszédesebb protokoll, mint az SMTP, gondoljuk tehát meg a választását. Ha az útválasztócsoportokat, telephelyeket X.25 köti össze, csak ez használható.
Kétirányú kapcsolat kiépítéséhez két egyirányú csatoló felépítése, beállítása szükséges.

A Link State Table

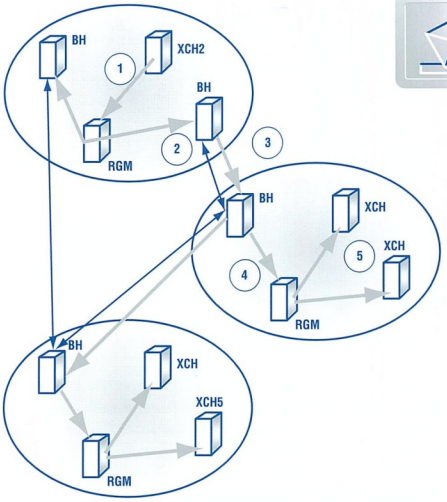
A szervezetben levő kiszolgálók mindegyikén található egy táblázat, ami az összes lehetséges utat, és annak árát *(cost)* is tartalmazza. A kiszolgálók mindegyike használja a táblázatot, útvonalat néz ki belőle, merre küldje az üzeneteket. A Link State Table tartalmazza a lehetséges utakat, a csatolók állapotát *(Up vagy Down)*, és az árat is. Az Exchange takarékoskodik, mindig a legolcsóbb utat választja az üzenetek továbbításánál. A Link State táblázat minden kiszolgálón megtalálható. De hogyan terjeszti az Exchange ezt a táblázatot a kiszolgálók közt...?

A Routing Group Master

Amikor szétosztjuk a kiszolgálókat a különböző útválasztócsoportokba, az első kiszolgáló lesz a főnök *(Master)* a csoportban. A Routing Group Master elsődlegesen az útválasztásért felelős, ő az, aki a Link State tábla frissítését és terjesztését végzi.

A Link State táblázat terjesztése

Mindig, minden kiszolgáló résen van, és figyeli a csatolók állapotát. Amint egy csatolón keresztül nem tud levelet küldeni, rövid időn belül megváltoztatja a saját Link State tábláját, és a változásokról értesíti a saját csoportjában levő Routing Group Mastert. Mindez a 691 TCP porton keresztül történik, SMTP protokollal, de nem levélben. A Routing Group Master feladata egyrészt, hogy a saját csoportjában levő egyéb kiszolgálókat értesítse, másrészt a Bridgehead kiszolgálókon keresztül eljuttatja a megváltozott táblázatot a többi útválasztócsoportba. A csoportok közt a Link State tábla a 25-ös TCP porton keresztül jut el, de nem SMTP üzenet formájában, hanem közvetlen a X-LINK2STATE SMTP paranccsal.



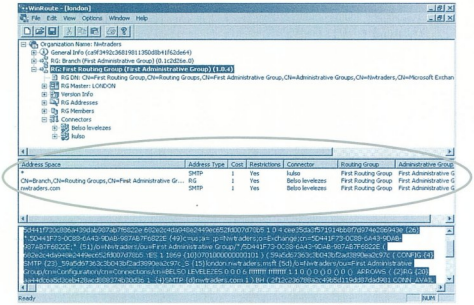
Lássunk egy konkrét példát a Link State tábla-változás szétkürtölésére a fenti ábra segítségével:
Az XCH2 szeretne levelet küldeni XCH5-nek. Igen ám, de a két útválasztó közti kapcsolat éppen nem működik, és ezt az XCH2 észre is veszi.

1. Az XCH2 értesíti erről a saját Routing Group Masterét, persze előtte a saját Link State tábláját frissíti
2. A Routing Group Master is megváltoztatja a Link State táblát, és elküldi mindenkinek a saját útválasztócsoportjában.
3. A Bridgehead kiszolgáló elküldi a szomszédos útválasztócsoportokba a frissített táblát.
4. A Bridgehead kiszolgáló frissíti a saját tábláját, majd elküldi a táblát a Routing Group Masternek.
5. A Routing Group Master szétkürtöli a saját csoportjában.

A változás végiggyűrűzik az összes útválasztócsoporton, előbb-utóbb mindenki értesül a változásról. Az egész folyamat elég gyors, a legelső Routing Group Master már 5 perccel a változás után tud róla.
Van pár eset, amikor a linkek sosem váltanak át Down állapotba. Ezek a következők:

- SMTP-csatoló esetén ha DNS alapokon *(az MX rekordok segítségével)* kommunikálunk.
- Ha Routing Group-csatolón keresztül bármelyik kiszolgáló küldhet levelet, vagyis ahhoz a csatolóhoz helyben mindenki Bridgehead.

A WinRoute nevű programmal *(amely az Exchange 2000 telepítő CD-n található)* a Link State tábla tartalmát belülről is meg lehet nézni.



WinRoute – a Link State tábla tartalma

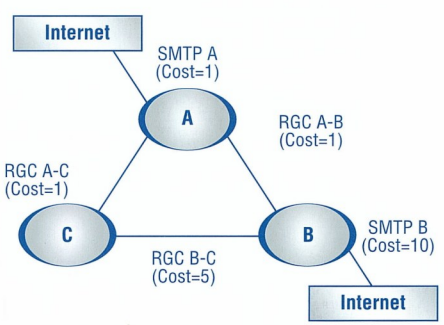
WinRoute segítségével a szervezetben bármely távoli kiszolgáló Link State táblája megtekinthető, nem csak a helyi kiszolgálóé.

Útválasztás

Végül rakjuk össze a kirakó darabjait!
Van több útválasztócsoportunk, mindegyikben Exchange kiszolgálókkal. Ha a Link State tábla alapján egy kiszolgáló többfelé is küldheti a levelet, mégis mi alapján dönti el, merre indítsa?
Az Exchange útválasztását befolyásoló tényezők:

- Az útba eső linkek mindegyike élő-e – ezt a Link State tábla alapján dönti el az Exchange.
- Az útba eső összes link beállítása engedélyezi a levél továbbítását *(például ha egy csatolón a küldhető levélméretet korlátozzuk, ugyanakkor a küldendő üzenet nagyobb a megengedettnél)*.
- Ha a névtér alapján két lehetséges csatoló is rendelkezésre áll, amelyen mehetne az üzenet, akkor a legjobban hasonlítót fogja választani a kiszolgáló.
- Merre a „legolcsóbb" elküldeni az üzeneteket – A Link State tábla „Cost" oszlopából számolja ki az Exchange.

Megpróbálom konkrét példákon szemléltetni az útválasztást és a levéltovábbítást.



Az ábrán három telephelyet láthatunk: **A, B** és **C**. Ha minden kapcsolat élő, a levéltovábbítás elve az, hogy a két Internetkijárat ellenére elsősorban az **A** útválasztócsoporton keresztül menjenek ki a levelek, de baj esetére rendelkezésre áll a **C** csoportban is egy SMTP-csatoló az Internet felé. A szervezeten belüli levélforgalomra pedig az jellemző, hogy **B** és **C** útválasztócsoportból csak **A**-n keresztül mehetnek a levelek, kivéve, ha **A**-ban valami baj van.

Első példa

Tegyük fel, hogy **C**-ből szeretnénk **B**-be levelet küldeni.
A **C** csoportban levő kiszolgáló megvizsgálja a lehetséges utakat:

- Küldhetné rögtön **B**-nek, hiszen van csatoló, annak ára 5 fitying.
- Küldheti **A**-nak, majd **B**-nek két csatolón keresztül, melynek ára 1+1 azaz 2 fitying.
- Tegyük fel, hogy mindegyik kapcsolat élő, és nincs korlátozás egyiken sem.

Az Exchange takarékos, a legolcsóbb utat fogja választani, tehát az **A csoporton keresztül** fogja elküldeni a levelet **B-nek,** hisz az csak 2 fityingbe kerül, míg a másik 5 fitying.

Második példa

Most **A**-ból **B**-be szeretnénk üzenetet továbbítani, de az **A** és a **B** közti **RGC A-B** épp nem működik.
Az **A** csoportban levő kiszolgáló a következő módon „gondolkodik":

- Küldhetné a levelet közvetlenül **B**-nek, ez egyetlen fityingbe kerül.
- Küldhetné az üzenetet először **C**-nek, majd **B**-nek, ez 1+5=6 fityingbe kerül.
- Megnézi a linkek állapotát, azt találja, hogy az **RGC A-B** csatoló nem működik, de nincs egyéb korlátozás a csatolókon.

A megszerzett információk alapján csak egy irányba indíthatja az üzenetet, igaz hogy az drágább – 6 fitying –, de nincs mit tenni, **C** felé kell küldeni az üzenetet. Nincs más út.

Harmadik példa

B-ből szeretnénk az Interneten keresztül külső címre levelet küldeni.
A **B** csoportban levő kiszolgáló a következő módon jár el:

- Küldhetné a levelet rögtön a saját SMTP-csatolóján keresztül, ez 10 fityingbe kerülne.
- Küldhetné az **A** csoporton keresztül, mert ott is ván SMTP-csatoló, ez 2 fityingbe kerülne.
- Küldhetné **C** csoporton keresztül **A**-ba, majd onnan az Internet felé, ez 7 fityingbe kerülne.
- Megnézi, mely csatolókon talál korlátozásokat és a Link State táblából megnézni, mindegyik csatorna működik-e. Tegyük fel, hogy minden rendben.

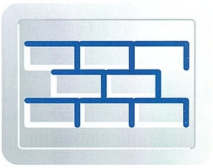
Az Exchange megint csak az ár alapján dönt, az olcsóbb úton indítja el az üzenetet, vagyis az **A** csoporton keresztül megy ki az Internetre, mert az a legolcsóbb, 2 fityingbe kerül.

Ez a módszer az OSPF útválasztáshoz hasonlít. Tanulság, hogy sokszor az útválasztás nem is olyan egyszerű, és ésszerűen beállított csatolókkal hibatűrőbbé tudjuk tenni a levelezési szolgáltatást.

Folytatjuk…

Dorner Csilla
dorner.csilla@netacademia.net

Eltemetett bitek: szteganográfia



Avagy nem minden arany, ami fénylik?

„...ellentétben a kriptográfiával, ahol a támadó észreveheti az üzenetet és módosíthatja azt, a szteganográfia célja, hogy a nyílt szöveget úgy rejtse el a gyanúmentes üzenetbe, hogy a támadó ne is láthassa meg, hogy a továbbított üzenet egy második üzenetet tartalmaz."

Napjainkban egyre többször kerül szóba a titkosítás, mint a biztonságos üzenettovábbítás egyetlen módja. Ha valamit el akarunk rejteni a kíváncsi szemek elől: titkosítunk. De tényleg ez az egyetlen mód az információ védelmére?

A szteganográfia

Van egy másik módszer is a titkos üzenetváltásra, amikor nem titkosítunk, csak elrejtünk valamit valamibe. Az eljárás nem újdonság, különböző megvalósításait az ókoriak éppúgy használták, mint a XX. század embere a II. Világháborúban. A görög származású szteganográfia szó jelentése: rejtett írás. Ez a szó sok eljárást foglal magába, melyeket a védendő kommunikáció során használhatunk, ilyenek például a láthatatlan tinta, mikroírás, betűk és szavak eltolásos elhelyezése és ismert kommunikációs csatornák közé titkosak keverése és így tovább. Markus Kuhn 1995-ben ezt írta az eljárásról: „A szteganográfia a kommunikáció művészete és tudománya, lehetőség magának a kommunikációnak az elrejtésére. Ellentétben a kriptográfiával, ahol a támadó észreveheti, feltörheti és módosíthatja az üzenetet, a szteganográfia célja, hogy a nyílt szöveget úgy rejtse el a gyanúmentes üzenetbe, hogy a támadó ne is láthassa meg, hogy a továbbított üzenet egy második – esetleg titkosított – üzenetet tartalmaz". Vagyis a titkosítás az üzenet értelmét változatja meg, míg az adatrejtés az információcsere tényét igyekszik titkolni, elfedni. Ha nincs titkosításra utaló jel, a feltörés provokációja is kisebb. A szteganográfiát egyes *(különösen katonai)* irodalmakban átviteli biztonságnak *(transmission security, TRANSEC)* is nevezik. A magyar szóhasználatban a lényeget jól megmutató kifejezés, az adatrejtés *(data hiding)* terjedt el.

Néhány legenda...

Áttekintve a történeti emlékeken láthatjuk, hogy az információ elrejtésére számtalan módszert dolgoztak ki az idők folyamán. Az ókori Görögországban viasszal bevont táblákat használtak az írásra. Egy történet szerint, amikor Demeratus figyelmeztetni akarta Spártát, hogy Xerxész Görögország inváziójáját tervezi, lekaparta a viaszréteget, és üzenetét a csupasz fa rétegre írta fel. Ezután az egészet úgy vonta be ismét viasszal, hogy egyetlen ellenőrzés során sem derült fény a rejtett üzenetre. Egy másik, nem igazán emberbarát módszer volt, hogy egy megbízható rabszolga fejét leborotválták és az üzenetet a csupasz fejbőrre tetoválták. Miután a küldönc haja ismét megnőtt, elindulhatott és az üzenet mindaddig láthatatlan maradt, amíg meg nem borotválták ismét.

A láthatatlan írás egyszerűbb és szokásos módszere volt a láthatatlan tinták használata, amely igen jól helyt álltak a II. Világháborúban is. Egy ártatlannak tűnő levél sorai között sok fontos információt rejthettek el így. A világháború korai szakaszában az alkalmazott szteganográf módszerek köre szinte kizárólag a láthatatlan tinták használatára korlátozódott. Ezek általában tej, ecet, gyümölcslevek felhasználásával készültek és közös jellemzőjük, hogy száradás után láthatatlanok lesznek, de melegítésre elsötétednek vagy egy másik anyaggal kezelve válnak láthatóvá. A technológia fejlődésével a láthatatlan tinták láthatóvá tétele egyre könnyebb lett, ezért sokan kísérleteztek azzal, hogy egyre többféle kémiai anyag felhasználásával egyre biztonságosabb tintákat állítsanak elő.

A kommunikációs csatornákon a sok titkosított üzenet között gyakran kódolatlan *(null ciphers)* szövegek is közlekedtek, de korántsem biztos, hogy minden az volt, aminek látszott. Az alábbi angol nyelvű szöveg egy időjárással kapcsolatos jelentés:

```
News Eight Weather: Tonight increasing snow.
Unexpected precipitation smothers eastern towns.
Be extremely cautious and use snowtires especially
heading east. The highways are knowingly slippery.
Highway evacuation is suspected. Police report
emergency situations in downtown ending near
Tuesday.
```

Ha fondorlatos módon csak a szavak első betűjét olvassuk, ezt kapjuk:

```
Newt is upset because he thinks he is President.
```

Ahogy egyre több módszert dolgoztak ki és alkalmaztak a titkos üzenetet váltani kívánók, a cenzorok egyre szélsőségesebb módon próbálták ellátni a feladatukat: tilos volt például olyan házhozszállítási megbízásokat postázni, amelyen *(kézbesítési)* időpont szerepelt, tilos volt keresztrejtvényeket, rejtvényeket és általában minden olyan dokumentumot postai úton elküldeni, ami titkos üzenetet tartalmazhatott. Ha a cenzor végképp nem tudott „belekötni" egy levél vagy képeslap tartalmába, akkor összecserélgette a bélyegeket vagy a szavak sorrendjét, esetleg egyes szavakat más, rokon értelmű szavakkal helyettesített vagy éppenséggel újrafogalmazta az egész levelet. Minden új – az üzenet elrejtését célzó módszer – már meglévő eszközöket és ötleteket használt fel és alapvetően nem sokat változott. Az igazi újdonság az volt, amikor a régi módszer alapötletét ötvözték egy újjal: az üzenetet változó vonalak, színek és egyéb képi elemek felhasználásával kódolták.

Szteganográfia ma

Az elektronika, a digitális technika fejlődése új utat nyitott a szteganográfia számára, és ma is aktív kutatási területnek számít, főként a szerzői jogvédelem és a rejtett csatornák alkalmazásának és felderítésének területén. Talán már az eddigiekből is nyilvánvaló, hogy a módszer lényege nagyjából az, hogy nagy tömegű „lényegtelen" információ között elrejtjük a védeni kívánt üzenetet.

Terminológia

A továbbiak során azt, amibe beletesszük az információt hordozónak *(cover, covertext)* nevezzük, amit beleteszünk, azt egyszerűen üzenetnek *(plaintext)*. Amit eredményül kapunk, az a stegotext. Hogy az üzenet nyílt szöveg vagy pedig rejtjelzett, az eljárás szempontjából közömbös. A „digitális szteganográfia" legtöbbször a hordozó egyes bitjeinek, bitcsoportjainak a megváltoztatását jelenti, így mielőtt elkezdenénk az adatmanipulációt, egy feltételt el kell fogadnunk: csak olyan adat lehet hordozó, melynek értelmezésében nem okoz zavart, ha leírásához használt bájtok egyes bitjeit, bitcsoportjait, *(például a legkisebb helyértékű bitjeit)* megváltoztatjuk. Az adat feldolgozásának eredményében a megváltozott bitek általában egyfajta zajként jelennek meg: minél több bitet változtatunk meg, annál erősebben. Egy jó adatrejtő rendszerről is fel lehet tételezni, hogy a támadó – hasonlóan a titkosító rendszerekhez –, a rendszer minden elemét ismeri, kivéve a feldolgozást esetleg vezérlő titkos kulcsot. A továbbiakban ezt a tulajdonságot nem veszük figyelembe, de gyakorlati megvalósítás során erre emlékezzünk!

Más rokon jellemző is van az adatrejtés és a titkosítás között: a lehetséges támadási módok. Az adatrejtés ellen irányuló támadásoknak a kommunikációs csatornák elleni támadásokhoz hasonlóan két fő módszere van:

- az adatrejtés elleni passzív támadásnak azt nevezzük, amikor a támadó felfedi az adatrejtés tényét, az elrejtett adatot esetleg megismeri, de nem változtatja meg vagy nem tudja megváltoztatni.
- az adatrejtés elleni aktív támadásnak nevezzük, ha a támadó az elrejtett adatot nemcsak megismeri, hanem azt megváltoztatni vagy eltávolítani is képes.

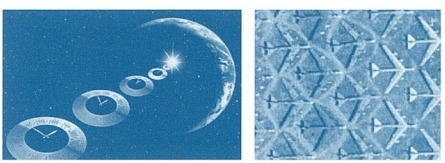
Az adatrejtés célja

Alapvetően két fő irányt különböztetünk meg az adatrejtési technikákban. Az egyik esetben nem kritikus a „rejtettség", viszont fontos, hogy az elhelyezett adat robosztusan, lehetőség szerint eltávolíthatatlanul kerüljön be a hordozóba. Ki kell állnia azt is, ha a stegotextet az általánosságban használt transzformációknak vetik alá. A másik esetben nem annyira fontos a robosztusság, viszont garantálni kell a rejtettséget, az észrevétlen továbbítás lehetőségét. A gyakorlatban ezek mellé még egy jellemző társul: az elrejtendő adat mennyisége. Ahol az eltávolíthatatlanság a fő követelmény, ott általában kevés adatot kell elhelyezni, ahol pedig a rejtettség, ott viszonylag sokat. Némi ellentmondás, hogy minél több adatot kell elrejteni, annál kevésbé valósítható meg a rejtettség. Az egyensúly az adott feladattól és alkalmazástól függ. A hordozó, mint adattovábbító csatorna kapacitását az az információmennyiség jellemzi, amely sikeresen átvihető és elő is állítható a vételi oldalon – felfedezés nélkül.

Példa: Image steganography – adatrejtés képbe

A mai digitális adattengerben igen sok lehetőség van arra, hogy elrejtsünk valamit valamiben: az ISDN csatornán folyó telefonbeszélgetéstől és adatforgalmaktól kezdve a különböző hang, kép és videó-formátumokig, vagy szinte bárhol, ahol digitális adatok közlekednek. Igazán azok a digitális csatornák alkalmasak, amelyek valamilyen emberi információt továbbítanak: hangot vagy képet, mert ezek – természetükből adódóan – tartalmaznak bizonyos redundanciát, látszólag felesleges adatot. Az ilyen „hordozó-jelölteket" digitalizálással készítik, ennek lényege, hogy a továbbítandó hangot, fényerőt – vagy egyéb jelet – úgy alakítunk át, hogy elektromos úton mérhető legyen. Ezután bizonyos időközönként mintát veszünk a jelből, és az abban a pillanatban mért értéket digitális eszközökkel valahány biten ábrázoljuk *(általában 8-24 biten)*. És itt máris adódik egy lehetőségünk az adatrejtésre! Ha ugyanis a számábrázolás a kettes számrendszer szabályainak megfelelően történik, a legkisebb helyértékű bitek esetleges megváltozása nem lesz jelentős hatással a rögzített jel rekonstruálására. Gyakorlatilag egy kissé pontatlan mérésnek fog tűnni. Természetesen nem szabad túl sok bitet módosítani, mert a változás előbb-utóbb akkora lesz, hogy az már zavaró és észrevehető. Teljesen azonos elvek szerint helyezhetünk el információt digitalizált fényképekben is, jóllehet azok készítésének folyamata kicsit más. A szteganográfia alkalmazásakor azonban mindegy, hogy az adat honnan származik, csak az adatok felépítéséről kell sok mindent tudnunk annak érdekében, hogy az eredeti információ minél kisebb sérülést szenvedjen. Lássunk egy példát:

> Minél több adatot kell elrejteni, annál kevésbé valósítható meg a rejtettség.



B-52-es bombázók a Davis-Monthan Air Force támaszponton

Ki hinné, hogy a bal oldali dizájnos kép a jobb oldali légifelvételt is tartalmazza? Erről a **[B52]** címen győződhetünk meg.

Az adatokat általában valamilyen feldolgozás után küldik tovább, ekkor még a feldolgozás előtt el kell rejtenünk az információt. Gondoljunk egy hangfelvételre: sok a szünet, a hosszabb-rövidebb csönd, vagy ha ránézünk egy fényképre igen sok, viszonylag homogén területet találhatunk rajta. A továbbításra alkalmas formátumok túlnyomó többsége eltünteti ezt a redundanciát *(tehát tömörít)* és olyan formába alakítja a digitalizálás közvetlen eredményeit, amelyek már emberi szemmel, füllel nem értelmezhetőek és többé-kevésbé érzékenyek az adatsérülésre is. A formátumtól függően elképzelhető, hogy még ekkor is van lehetőség adatmanipulációra, kihasználva az adott formátum specialitásait *(például kitöltőrészek, metainformációk, extension tags stb.)*.

LSB módszer 1.

A korábbi feltételeknek jellemzően megfelel a kódolatlan, truecolor BMP formátumú képállomány. Ez a fájlformátum a képet

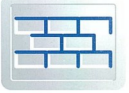
úgy tárolja, hogy minden egyes mintavétel és mérés eredménye tömörítés vagy más varázslat nélkül kerül az állományba. Gyakorlatilag a mérési eredmények szekvenciális halmaza. Ez a formátum megengedi a pixelek színmintáinak kismértékű megváltoztatását, amit az emberi szem csupán zajként érzékel. A változtatások a BMP formátumot nem teszik tönkre. Ha kevesebb bitet használunk fel a mintákból, a zaj kevésbé lesz észrevehető, ha többet, erősebb lesz. Általában nem alkalmasak hordozónak a szövegfájlok, alfanumerikus adatokat tartalmazó táblák, vektorizált térképek, tömörített vagy bináris állományok.

A fentiek figyelembevételével tekintsünk egy true-color 1024x768 pixel méretű bitképet egyelőre tömörítés, fejléc és más formátum információ nélkül:

- Minden képpont leírására 3 bájtot használunk, ezek rendre a vörös, zöld, kék összetevőket tartalmazzák. A kép mérete ekkor: 1024x768x3x8 = 18 874 368 bit. *(18Mbit, 2304Kb)*
- Ha minden képpontból „ellopjuk" a legalsó bitet és helyére az üzenetet írjuk, a kép alapvetően nem változik meg, a színek megváltozását emberi szemmel nem lehet észrevenni.
- Egy bit felhasználásával 1024x768x3x1 = 2 359 296 bit *(2.25Mbit, 288Kb)* adatot tudunk elraktározni a hordozó képbe, annak 12,5%-át felhasználva.
- Lehetőség van arra is, hogy ne 1, hanem 2, 3 vagy 4 bitet „csípjünk" le. Ekkor rendre 576Kb *(25%)*, 864Kb *(37,5%)* vagy 1152Kb *(50%)* kapacitást biztosít a hordozó. Minél több bitet használunk el, annál nagyobb lesz a színek torzulása, amit előbb vagy utóbb már emberi szemmel is észre lehet venni.

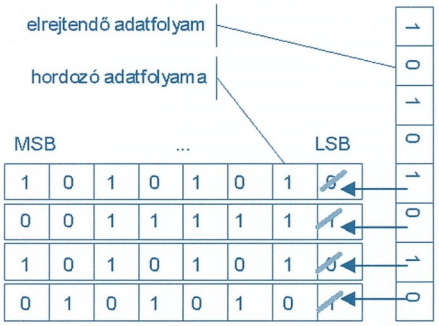
Az ilyen nagyfelbontású, RGB színsémát alkalmazó, tömörítetlen képek egyik legnagyobb baja, hogy hatalmasak. A példában szereplő kép fájlként legalább 2Mb, de még egy átlagosnak mondható 640x480x3 tulajdonságokkal rendelkező kép is 900Kb. Ha ilyen méretű állományt továbbítani akarunk, tömörítenünk kell. Ha magát a képfájlt, mint bináris állományt veszteségmentesen tömörítjük, elküldjük és a túlsó oldalon pedig kicsomagolják, semmi problémánk nincsen. Ha azonban a képet már eleve valamilyen tömörített formátumban akarjuk tárolni, nem mindegy, hogy milyen formátumot választunk. Egy a lényeges, hogy ne veszteséges *(például JPEG)* formátumot válasszunk, mert sajnos szembe kell néznünk az ilyen tömörítők sajátosságával: adatveszteség lép fel, ráadásul pont a legkisebb bitek helyén. Így tömörítéskor pont az az információ vész el, amit az előbb tuszkoltunk bele a hordozóba. Ezzel szemben nyugodtan használhatunk minden más olyan formátumot, amely ismeri a 24 bites színmélységet, emellett nem okoz adatveszteséget *(TIFF, PNG, JPEG2000)*.

Főleg az internetes alkalmazásokhoz szívesen használják a 256 szín kezelésére képes GIF formátumot. A formátum jó tömörítést biztosít a kisebb képekhez és viszonylag egyszerű a kódolása is. A gond csak az, hogy itt egy pixelérték csak közvetetten jelenti a pont színét: csak egy mutató a palettába, ahol végül majd eldől a pont színe. Tehát nem lehetünk biztosak benne, hogy ha megváltoztatunk egy 11110000_2 értékű pixelt mondjuk 11110001_2-re, akkor senki semmit nem fog észrevenni, mert lehet, hogy az első érték fekete színre, a második érték pedig fehérre mutatott. Csak akkor alkalmazhatjuk a már vázolt módszerünket, ha a palettában az egymást követő színek hasonló RGB értékekkel rendelkeznek, vagy addig cserélgetjük a pixelértékeket és a hozzájuk tartozó palettabejegyzéseket, amíg hasonlóak nem lesznek. Átmeneti megoldást jelenthet a palettaalapú színábrázolásokban a szürkeskálás képek alkalmazása, ahol csak egy bájt ír le egy pixelt, de egy-két bitet itt is fel tudunk használni. E képeknél szinte kizárólag lineáris megfeleltetés van a 0-255 pixelértékek és a fekete-fehér átmenet között.



LSB módszer 2.

A képek területén a BMP olyan, mint a hangfelvételek esetében a PCM formátumú hangfájl. Ez a fájlformátum a digitalizált hangot szintén a mérési eredmények időrendes, szekvenciális halmazaként tárolja. *(Lényegében ugyanilyen az audio CD formátuma is, csak ott még különböző hibajavító kódok és technikák is szolgálják a felhasználókat.)* Ez a „nyers" formátum megengedi a hangminták legkisebb helyértékű bitjeinek megváltoztatását, amit az emberi fül csupán zajként érzékel. A változtatások a PCM formátumot nem teszik tönkre. Ha kevesebb bitet használunk fel a mintákból, a zaj kevésbé lesz észrevehető, ha többet, a zaj erősebb lesz. A PCM kódolású hangfájlok meglepően sok kapacitást képesek biztosítani. Ha az alsó négy vagy akár nyolc*(!)* bitet felhasználjuk, a WAV fájl méretének 25% illetve 50%-a áll rendelkezésre, ami már több megabájt adat továbbítását is lehetővé teszi, például egy audio CD-n. Egy négyperces – átlagos hosszúságú – sztereó WAV 44,1 kHz mintavételezéssel, 16 bites mintákkal 4x60x2x44100 = 21 168 000 mintát = 42 336 000 bájtot jelent. Ha itt elhasználunk 4 bitet – a bátrabbak nyugodtan próbálkozhatnak akár 8 bittel is –, akkor 21 168 000 x 4 = 84 672 000 bit, azaz megközelítőleg 10 Mbájt információt tudunk elrejteni.



LSB módszer: a hordozó kevésbé fontos bitjeit egyszerűen kicseréljük ...

Jóllehet az LSB módszer igen sok adat átvitelét teszi lehetővé *(high payload)*, a módszer – ismertsége miatt – önmagában már nem alkalmas biztonságos adatrejtésre.

Interneten általában nem is .wav, hanem MP3 fájlokat továbbítunk, azok erőteljes tömörítési aránya miatt. Arra is van lehetőség, hogy MP3-ba rejtsünk adatokat **[mp3stego]**.

A kivétel erősíti a szabályt: szöveges állományok

Az alapelvek tisztázásakor elfogadtuk, hogy csak olyan adat alkalmas hordozónak, amelyik elviseli, ha egyes bitjeit megváltoztatjuk. Azonban más szempont szerint is rejthetünk el információt. Ha azt mondjuk, hogy az alkalmas hordozónak, ami elviseli, hogy az adatok közé pótlólagos információt szúrunk

be, akkor lehet, hogy egy szöveges állomány is alkalmas adatrejtésre. Írjuk fel a rejteni kívánt adatokat egyetlen bitfolyamként! Hordozónak válasszunk egy jó hosszú szöveget, például egy novellát!

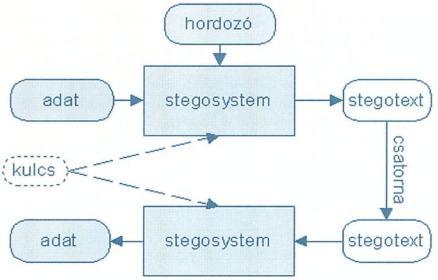
Végezzük el a feldolgozást a következőképpen:

- A hordozószöveget soronként olvassuk be.
- Ha a sor végén egy vagy több szóköz karakter található, töröljük ki mindet onnan.
- Ezután vegyük a rejteni kívánt bitfolyam következő – mondjuk – három bitjét bináris számként.
- Ennek megfelelő mennyiségű szóközt szúrjunk a sor végére: ha például a bitek a 0112=310 számot adják ki, 3 darab szóközt adunk a sorhoz.

Minél hosszabb bitcsoportokat használunk, minél több szóközt szúrunk a sor végére, annál nagyobb a „lebukás" veszélye, de annál több információ tárolható egy adott szövegben. Minél kevesebbet, annál nagyobb a biztonság, de ezért cserébe kevesebb a tárolásra használható hely. Számszerű példa: az A4 méretű oldalon átlag 55 sor van, ez az iménti 3 bitet feltételezve 55x3=165 bit=20 bájt elrejtését teszi lehetővé oldalanként. Ez nem sok, de néhány módszer még ennyit sem képes elrejteni. Igaz, ezek másban jeleskednek, például az elrejtett információ nem változtatja meg a hordozó statisztikai jellemzőit, vagy éppen nagyon sok mindent képesek átvészelni. Más jellegű átalakításokkal is érhetünk el eredményeket. Ha megnézzük az olyan szöveges fájlokat, mint a HTML vagy az RTF formátum, egy kis ötlettel ott is rejthetünk el információt úgy, hogy a leírt dokumentum megjelenítésekor semmit nem veszünk észre. És nem csak arról van szó, hogy nem vesszük észre a változásokat, hanem hogy a megjelenített vagy kinyomtatott dokumentum, az eredetivel pontosan egyező lesz! Ugyanis mindkét formátumnak az a közös jellemzője, hogy olyan leírónyelvet használnak, amely angol szavakon vagy azok rövidítésein alapul, és a dokumentum fizikailag tiszta, hétbites ASCII formátumban leírható. Mindkét nyelvben egyértelműen elkülöníthetők a formázáshoz használt kulcsszavak a dokumentum szövegétől vagy egyéb objektumától: a HTML-ben „<" és „>" jelek, a RTF-ben „{" és „}" jelek határolják őket. Egyik formátum sem érzékeny arra, hogy ezek a határolt egységek kis- vagy nagybetűvel vagy éppen vegyesen vannak írva. Ezért akárhogy írhatjuk őket: ahol egy „1"-es bitet akarunk jelezni, oda nagybetűt írunk, ahol „0" a kérdéses bit, oda kisbetűt. Így például a „<HTmL>" tag egy 11012 bitsorozatot képviselhet. Sajnos nem egészen ilyen egyszerű a helyzet, mert a HTML-ben Javascript is lehet, ami viszont érzékeny a kis és nagybetűk közötti különbségre. Hasonlóan nem írhatjuk át az „<img>", a „<form>", a „<link>" és „<a>" tagok fájlhivatkozásait sem, mert az UNIX-os szerver esetén katasztrofális lehet. A korlátokat figyelembe véve viszont szabadon garázdálkodhatunk. Ismereteim szerint RTF-ben hasonló korlátozások nincsenek, csak a fájl első hat karakterének kell „{rtf1}"-nek lennie *(kisbetűvel)*. Az eljárás során a hordozó mérete nem változik meg.

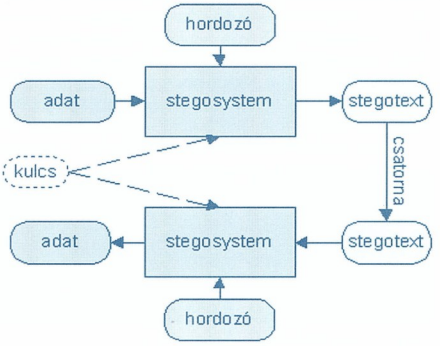
Szimmetrikus és aszimmetrikus adatrejtés

Az eddig bemutatott szteganográf eljárásoknál nincs szükség az eredeti hordozóra az elrejtett adat kinyeréséhez, mert anélkül is dekódolható az elrejtett adat, így azt nem kell továbbítani, vagy előzetesen egyeztetni. Ezeket az eljárásokat vak eljárásoknak nevezzük *(blindly methods)* és a jelenleg használt technikák döntő többsége ilyen.



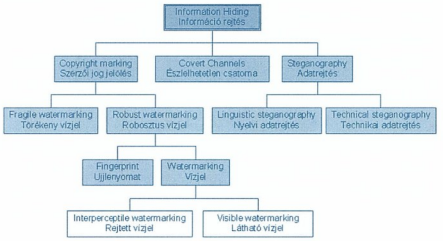
Blind eljárás

Egy egyszerű példa a másik megoldásra: az elrejtendő adatbiteket egy képben úgy helyezzük el, hogy az egyes biteket valamilyen szabályszerűség szerint az eredeti kép fölé „terítjük", majd az eredeti pixelértékekhez hozzáadjuk *(valamelyik színcsatornához, de akár mindháromhoz egyszerre)*. A létrejövő színingadozás egy részletdús képnél senkinek sem fog feltűnni. Az ilyen eljárásokat letételjárásoknak nevezzük *(cover escrow)*.



Cover escrow rendszer

Jóllehet ez utóbbi eljárások biztonságosabbak, azonban a gyakorlati megvalósítás során jelentős problémát okozhat az eredeti hordozó eljuttatása a címzetthez, vagyis ugyanazok a problémák jelentkeznek ismét, amelyek a szimmetrikus titkosító algoritmusok kulcscsere problémáját is jellemzik.



Információrejtési technikák a felhasználás célja szerint

Alkalmazási területek

A fenti ábra a főbb technikák csoportosítását mutatja, melyek más-más felhasználási területhez tartoznak, és különböző a megvalósítandó céljuk is. A következőkben ilyen szempont szerint ejtünk néhány szót az egyes megoldási lehetőségekről.

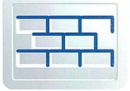
Copyright watermarking

A szteganográfia egyik gyakori felhasználása az elektronikus úton terjesztett publikációk védelme, az elektronikus vízjel- vagy ujjlenyomat alkalmazása. A grafikával foglalkozók egy-egy elkészült képükbe például az Adobe Photoshop használatával tudnak egyedi azonosítót tenni. De hasonlóan lehetne minden audio CD sávba is copyright információt rejteni. Akár képről, akár hangról van szó, a módszer biztosítja, hogy az azonosító minden másolatban benne lesz, onnan gyakorlatilag eltávolíthatatlan. Ha copyright jelzésként nem szöveget vagy számot helyezünk el, hanem képet, emblémát vagy hangot *(tehát olyan üzeneteket, melyek egyébként is redundánsak így „sok mindent kibírnak")*, valószínű, hogy még különböző szűrések, „belerajzolások" után is megmarad a készítő eredeti*(hez hasonló)* azonosítója. Érdemes megemlíteni azt is, hogy ha a hamisító saját „védjegyet" tesz a „művére", az nem fogja az eredetit felülírni, hanem mindkettő megmarad *(főként ha különböző módon, különböző helyekre teszik azonosítójukat)*.
Az ilyen jellegű védelmek másik fajtáját, az elektronikus ujjlenyomatot *(fingerprint)* másra használják. Ennek feladata egy-egy másolat útjának nyomon követése, ugyanis az ujjlenyomat egy olyan egyedi azonosító, melyet minden egyes jogszerű másolatba elhelyeznek. Ha később egy illegális másolat felbukkan, a szerzői jog tulajdonosa az ujjlenyomat dekódolásával azonosíthatja az eredeti másolat tulajdonosát *(traitor-tracing, áruló-követés)*. A vízjelek között tehát két fő feladat oszlik meg:

1. Fragile watermarking – „törékeny vízjel": Feladata, hogy biztosítsa a legkisebb módosítás észrevételét is, mihamarabb felfedve így a hamisításokat, változtatásokat.
2. Robust watermarking – „strapabíró vízjel": Feladata, hogy túlélje a módosításokat, a lehető legtovább megmaradva bizonyítsa a megjelölt hordozó eredetét.

Covert channels

Az adatrejtés támogathatja az ún. észlelhetetlen csatorna *(hidden channel, covert channel)* megvalósítását, illetve biztonsági szempontból felveti ezek keresésének szükségességét. A kommunikáló felek az észlelhetetlen csatorna segítségével kiegészítő információkat küldhetnek egymásnak, például jelezheti a küldő a címzettnek, ha az adott dokumentumot egy kényszerítő harmadik fél hatására írta alá. A legfontosabb következménye a csatorna létének azonban az, hogy egy ilyen eljárást implementáló programozónak, hardver-gyártónak lehetősége van arra, hogy az észlelhetetlen csatorna segítségével olyan információkat juttasson ki egy rendszerből, amelyeket nem lenne szabad *(például egy kulcspár titkos kulcsát, jelszavakat vagy más érzékeny adatokat)*. Informatikai rendszerek biztonsági vizsgálatánál ezért az egyik legfontosabb szempont az észlelhetetlen csatornák keresése és megszüntetése.



Steganography

Az ábra harmadik főcsoportja azokat a módszereket takarja, amelyekről a cikk eddig is szólt. Ez nem más, mint adatrejtés tömeges adatátviteli célzattal, a kriptográfia alternatívájaként vagy azt kiegészítő módszerként. Ne feledjük, az adatátviteli szteganográfia fő célja, hogy úgy rejtse el az üzenetet, hogy a lehetséges támadó észre se vegye az üzenetküldés tényét, csökkentve így annak az esélyét, hogy egyáltalán megpróbálja dekódolni, feltörni az üzenetet. Más szavakkal ez azt jelenti, hogy az üzenet küldése nem provokálja az üzenet feltörését.

Titkosító módszerek vs adatrejtés

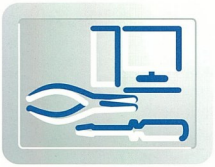
A szteganográfiai eljárásoknak rengeteg megvalósítása lehetséges és az eljárás nem csak digitális feldolgozásban használható, hanem egyéb médiákon is megvalósítható. A szteganográfiának helye van a titkosító rendszerek között, jóllehet azokat nem válthatja fel, de megvalósíthatja azok célkitűzéseinek egy részét.
Előnye a kriptográf módszerekkel szemben, hogy a továbbított üzenetnek kisebb esélye van arra, hogy a kódtörő egyáltalán észreveszi. Bár ha egy feltételezetten titkosított kommunikációban a felek csak képeket küldözgetnek egymásnak, az elég gyanús lehet. Ilyenkor célszerű a különböző módszereket keverni, vagy a csatornán valódi titkosított, de tartalom nélküli üzeneteket is küldeni.

Való világ

Néhány olvasó, aki a cikket megjelenése előtt olvasta, feltette azt a gyakorlatias kérdést, hogy a bankók vízjelét, zárjegyeket hova sorolnám? Nos, a zárjegy egyértelműen „törékeny vízjel". A bankók vízjele robosztus vízjel, és a papír eredetét igyekszik bizonyítani. Az egyéb ismérvek *(UV reagens szösök, hologram csík, stb.)* pedig a törékeny vízjelekhez tartozhatnak. Általában a legtöbb elhelyezett védjegy *(itt most nem a márkanevekre gondolok)* törékeny – így akadályozva meg az újrahasznosítást – és egyúttal az eredetiséget is igazolják.
Az egyértelmű megkülönböztetés vagy besorolás nehéz, mert legalább két fontos különbség van a fizikai formában megjelenő értékek és a fentebb tárgyalt technikák között:

1. Az elektronikus dokumentumoknál lényegtelen, hogy milyen fizikai formában tárolódnak. Egy bankónál viszont nagyon is fontos, hogy milyen papírból készült.
2. A bankóhamisítás során a különböző ismertető jegyek elhelyezése a cél *(bizonygatva, hogy a bankót az állam készítette)*, és nem azok eltávolítása *(eredet letagadása)*.

Virasztó Tamás
wacher@westel900.net



Portál Para[digma]

Avagy mit tegyünk, ha sürgősen portált kell építenünk?

Ha főnökeink vagy ügyfeleink azt várják tőlünk, hogy integráljunk egy-két tucat különböző alkalmazást úgy, hogy az információk és szolgáltatások az Interneten is elérhetők legyenek, mindezt egyszerű legyen kezelni, és persze az egész minimális költségvetésből és tegnapra kell, akkor bizony nehéz dolgunk van. Az ma már nem kérdés, hogy ilyenkor vállalati portált kell építeni, de hogy milyen eszközökkel, milyen architektúrával és milyen platformon, az már nem olyan egyértelmű. Jön a portál para.

Immár majdnem öt éve foglalkozom portálfejlesztéssel Microsoft platformon, és úgy érzem ez az üzletileg igen fontos téma méltatlanul kevés helyet kap a magyar szakirodalomban. Ezt a hiányosságot pótolandó kezdtem megírni ezt a cikket, amely terveim szerint egy cikksorozat első, bevezető része.

Mivel a vállalati portál egy erősen üzletorientált IT megoldás, nem hagyhatom ki cikkemből annak üzleti vonatkozásait. Annál inkább nem, mert a portálprojektek nagy része sajnálatos módon pont azért bukik meg, mert az üzleti és IT célok nem találkoznak. Az IT ugyanis nem mindig érti, mit kíván a menedzsment, az üzleti vezetők pedig nem ismerik a lehetőségeket, és még ha meg is kérdeznek minket, nem értik mit zagyválunk mi műszakiak.

Ebben a megoldhatatlannak látszó helyzetben azonban nekünk *(műszakiaknak)* van szerencsénk, ugyanis sokkal könnyebb némi „közgáz"-t tanulnunk, mint egy átlagos menedzsernek megérteni, miért hagyta ki a Microsoft a C#-ból a többszörös öröklődést.

Ebben az első cikkben szeretnék tehát némi üzleti szemléletet is átadni, de megpróbálom mindezt nem a műszaki részletek terhére tenni. A soron következő cikkek az itt felvetett témák műszaki részleteit fogják alaposabban kivesézni.

Üzleti probléma

A nagyobb szervezetek rendelkeznek különféle üzleti szoftverekkel, illetve a gyorsan változó üzleti környezethez újabbak fejlesztése, vásárlása szükséges. A meglévő szoftverek egy része modern, egy része régebben vásárolt vagy házilag fejlesztett, de a leggyakrabban előforduló probléma az, hogy nem integráltak, sok esetben ugyanazokat az adatokat tartalmazzák redundánsan, nemegyszer inkonzisztensen.

Ebből következik, hogy ahhoz, hogy a felhasználó komolyabb üzleti összefüggéseket találjon, sőt néha ahhoz is, hogy egyszerű információkhoz férjen hozzá, több alkalmazást, több különböző szoftvert kell futtatnia egymás mellett vagy egymás után. Ez általában ablakok kavalkádját eredményezi, és inkább hátráltatja, mint segíti az információszerzést. A probléma igen általános, szinte minden szervezetet, vállalatot érint.

A különféle alkalmazások átírása oly módon, hogy azok egy rendszerként működjenek igen nehéz és hosszadalmas feladat, és gyakran nem műszaki, hanem gazdasági, jogi akadályokba vagy szervezeti ellenállásba ütközik. Az effajta integrációs próbálkozások igen ritkán gazdaságosak. Tovább nehezítheti a megoldandó feladatot az, hogy ennek ellenére általában futnak kisebb-nagyobb integrációs projektek, amik ritkán sikeresek, de annál inkább csökkentik az IT megoldások iránti bizalmat a menedzsment köreiben. Éppen ezért fontos, hogy amennyiben portál építésbe kezdünk, pontosan ismerjük, miféle fába vágjuk a fejszénket.

Mi az a portál?

Évekkel ezelőtt az Internet elterjedésével elindultak az első olyan honlapok, melyek egy adott közösséggel vagy témakörrel kapcsolatos információkat gyűjtötték egy helyre. Az efféle honlapok nem maguk kínálták az információt és szolgáltatásokat, hanem „csak" hivatkozásokat adtak az egyes forrásokhoz; gyakran nem is voltak többek tematikus linkgyűjteményeknél. Az ilyen honlapok belépőpontként, kapuként szolgáltak a releváns adatok hatalmas halmazához, innen ered a portál, azaz kapu elnevezés.

> ...ebből a megközelítésből hasznot lehet húzni

A portálok ugyanakkor saját maguk is nyújtottak szolgáltatásokat, amelyet egyetlen hivatkozott forrás sem nyújtott önmagában, ezáltal a portál további értéket teremtett. A kategorizálás vagy a keresőszolgáltatás minden portál alapszolgáltatásai közé tartozik, célja az információk célbajuttatásának felgyorsítása, ezáltal az információra éhes személy üzleti előnyhöz juttatása.

A gazdálkodó szervezetek vagy cégek hamar rájöttek, hogy ebből a megközelítésből hasznot lehet húzni, és az Internetes portálokhoz hasonlóan a céges Intraneten is elkezdtek portálokat építeni. Az így kialakult vállalati portálok kezdetben Internetes társaikhoz hasonlóan egyszerű HTML oldalak voltak, egy-két alapszolgáltatással kiegészítve.

Napjainkban a vállalati portálmegoldások nem próbálják meg a lehetetlent, nem próbálják az alkalmazásokat teljes mértékben integrálni, csak az alkalmazásokból kinyerhető információ megjelenését és a fontosabb alkalmazásfunkciókat teszik egy képernyőre, pontosabban egy weboldalra. A különféle portálszoftverek képesek arra, hogy egy weboldalként, egy böngészőablakban jelenítsenek meg több forrásból jövő adatokat vagy felhasználói felületeket. Ez az egy képernyő így kaput nyit a szervezet különféle szoftverrendszereiben tárolt információihoz, más szóval információs Vállalati Portált valósít meg.

Internet – Intranet – Extranet

Egy szervezet üzleti folyamataiban sokan vesznek részt. Házon belül a cég alkalmazottai, dolgozók, menedzserek, vezetők és

tulajdonosok, házon kívül pedig a nagyközönség, az állami és önkormányzati szervek, vásárlók, megrendelők, partnerek és beszállítók. A szervezet hatékony működéséhez elengedhetetlen, hogy ezek a csoportok hozzáférjenek a nekik szükséges információhoz és szolgáltatásokhoz, illetve képesek legyenek a szervezet számára fontos információt közölni, a rendszerbe juttatni. Az is fontos, hogy a szervezet irányítani és felügyelni tudja, hogy a megfelelő információ a megfelelő irányba terjedjen, ne kerüljön illetéktelen kezekbe, viszont eljusson azokhoz, akiknek igazán szükségük van rá.

Az Internetportál jól ismert megoldás a nagyközönség tájékoztatására, de a portálmegoldás igazi előnye, hogy nem csak a nagyközönséget tudja információval ellátni az Interneten keresztül, hanem saját munkatársait is egy belső hálózaton, az Intraneten keresztül. Az Intranet nem más, mint egy a szervezeten belül működő, az Internet technológiáit hasznosító hálózati rendszer, egy privát, mini Internet. Az Intranet könnyedén kiterjeszthető a szervezeten kívülre is, így az információáramlás a szervezet határain kívül is folyhat, de korlátozott és a szervezet által felügyelt módon. A szervezet határain túlmutató, de a nagyközönségtől elzárt hálózatot extranetnek nevezzük.

Nemzetközi kutatócégek legfrissebb felmérései szerint a cégek olyan megoldásokat keresnek, amelyek mindhárom megoldást egy közös infrastruktúrán képesek ellátni, jelentősen csökkentve a beruházási és üzemeltetési költségeket. Menedzser kollégáink ezt úgy mondják, hogy a modern portálnak nagyobb a ROI-ja és kevesebb a TCO. A ROI az angol Return on Investment, azaz megtérülés, a TCO a Total Cost of Ownership, azaz teljes üzemeltetési költség kifejezést takarja. Beszerzéskor alaposan el kell gondolkodni, hogy a közeli és távoli jövőben miként kívánja a szervezet kihasználni a portáltechnológia adta kommunikációs csatornákat. Ha tehát a feladat Internetportál készítése, érdemes előre tekinteni, és felhívni a menedzsment figyelmét, hogy ha most megfelelő szoftvert vásárolnak, később alacsony költséggel intranetet vagy extranetet is tudunk majd építeni. Fordítva is igaz, intranetépítés esetén is akad olyan információ, amit a cég weboldalán érdemes a nagyközönséggel megosztani, és később esetleg extranet portált is létrehozni ügyfeleknek vagy beszállítóknak.

Single Sign-On *(Egy jelszó mind felett)*

A jó vállalti portál megoldást adhat egy másik, az alkalmazásintegrációval kapcsolatos problémára, a jelszavak kezelésére. Általános probléma, hogy ahány szoftver üzemel a szervezetben, annyi felhasználónév és jelszó párt kell megjegyezni, és ha az egyik rendszerben megváltozik, vagy törlődik a jelszó, nincs olyan ember, aki kényelmesen eligazodna a jelszavak kavalkádjában. Sok helyen ez akkor is így van, ha már van Active Directory, hisz nem minden alkalmazás „tudja" az AD-t, és nem is mindet lehet átalakítani. Új munkatársak felvitele és törlése is komoly gondot tud okozni. Gyakori, hogy az új kolléga első heteit jelszavak utáni rohangálással tölti munka helyett, illetve régen elbocsátott kollégák még mindig rendelkeznek hozzáféréssel kulcsfontosságú adatokhoz. Egyes kutatások szerint a nagyvállalati betörések *(hackelés)* hátterében az esetek többségében bosszúra éhes, elbocsátott munkatárs segítsége áll. Erre a problémakörre nyújt megoldást a portál egyjelszavas megoldása, angolul Single Sign-On *(SSO)*. A portálra bejelentkezéshez csak egy jelszóra van szükségünk, ezzel hozzáférést nyerünk minden olyan információhoz és funkcióhoz, amit a portál prezentál nekünk, függetlenül attól, hogy az információ melyik rendszerünkből érkezik, és a funkció melyik alkalmazásból érhető el. Ez a megoldás a felvétel-elbocsátás körüli problémákat is megoldja, mert elég egy jelszót kiadni, vagy visszavonni.

Ez a problémakör biztosan egy önálló cikket kíván, elöljáróban azonban kitérek egy fontos dologra a portálokkal kapcsolatban. Nem szabad, hogy az egységes megjelenésű portál GUI-tól várjuk az SSO-hoz hasonló kihívások megoldását. Az ilyen esetekben ugyanis nem elég a felhasználói felületet, a prezentációt kiépíteni és egységesíteni, hanem mögöttes üzleti logikát is kell alkotni.

> A statikus HTML generálásáról érdemes elfeledkezni.

Azok a portálgyártók, akik azt hirdetik, hogy termékeik megoldják az SSO és hasonló problémákat, nem mindig teszik hozzá, hogy ez jelentős szervezési munkát is igényel, ugyanis a sok rendszer között igen komplikált módon kell szinkronizálni a felhasználók azonosítóit, és ez egyedi integráció nélkül nem oldható meg; a portálszoftverek csak eszközökkel támogatják a munkát. Aki végzett már AD-bevezetést, tudja, hogy legtöbbször nem a telepítés, hanem a felhasználók adatainak összeszedése az igazi kihívás; így van ez egy komolyabb SSO-háttérlogika felépítésénél is. Szerencsére vannak olyan kész szoftverek, mint a Microsoft Metadirectory Services, melyek a dobozból kivéve, külön fejlesztés nélkül is sok problémát megoldanak.

Dinamikus oldalak

A modern portálmegoldások a weboldalakat alkotó HTML-t mindig a letöltés pillanatában állítják elő valamilyen adatforrásból, adatbázisból vagy alkalmazásból. Ez még akkor is így van, ha az URL látszólag statikus HTML-re mutat. Ez a folyamat kicsit időigényesebb, mint egy egyszerű, előre megírt, más néven statikus HTML oldal letöltése, hiszen a HTML-t itt egy program állítja elő, ráadásul minden látogatónak egyedileg. A portáloldalak általában kicsi „dobozkákból" állnak, amelyek mind egy-egy alkalmazást képviselnek a GUI-ban. Ezek a webpartok, portletek, gadgetek, vagy placeholderek ráadásul több forrásból egyszerre, de egymástól függetlenül generálódnak, ezért az ASP.NET megoldásoknál is lassabb futást kell feltételezni. A portáloldalak felépítése eltér egy ASP-oldalétól, hiszen az egyes dobozkák mögötti kód más-más gyártó műve is lehet, melyeket forráskódszinten nem lehet egybeépíteni, ráadásul a felhasználók is képesek rakosgatni őket *(testreszabáskor)*. Az egy oldalra felpakolt dobozkák tartalma tehát minden letöltéskor személye szabottan kerül előállításra, ami a portál legnagyobb előnye. Ha viszont több ezer látogató használja a portált, és az oldal nagy része nem változik, a portál motor ugyanazt a feladatot *(pl. kigyűjteni a legfrissebb híreket)* sok ezerszer végzi el, ugyanúgy. Ez fölösleges terheli a szervereket, ami nem megengedhető. A statikus HTML generálásáról pedig érdemes elfeledkezni, mert egy közepes portál esetében is iszonyú kínszenvedést okozhat egy apró menedzseri igény „beszélése" a HTML generátorba.

Cache

Az átmeneti tár, angolul cache – az előző példánál maradva – a kigyűjtött híreket egy rövid ideig tárolja, így nem kell a szervereknek fölösleges munkát végezniük. Fontos, hogy olyan portálmotort és fejlesztőeszközt válasszunk, ami képes a dobozkák tartalmát egyenként cacheelni. Így az oldalon elhelye-

zett dobozkák egy része mindig teljesen friss és személyre szabott lehet, az olyan adatok, melyek ritkábban változnak, viszont bemennek az átmeneti tárba, és onnan kerülnek kiszolgálásra. Ha az oldalunkat percenként több ezren nézik, egyperces átmeneti tárolás is több ezer adatbázis-lekérdezéstől mentesíti a szerverrendszert, így jelentős összegeket lehet a hardverbeszerzésnél vagy a fejlesztésnél megtakarítani.

Tudom, ez eretnekségnek hangzik, de a jó cachefunkció lehetőséget ad arra, hogy a portálunk egyes részeit ne kelljen „kioptimalizálni". *(Ez olyan projekteknél lehet igen hasznos, ahol vészesen közeleg, vagy már rég lejárt a határidő.)* A menedzsment sokkal elégedettebb lesz egy olyan portállal, ami néha – amikor lejár a cache – lassú, de egyébként határidőre elkészül és működik. Ekkor már könnyebben kapunk időt a gyorsításra. A .Net framework cache támogatása és annak megfelelő használata szintén egy tervezett cikk témája.

Skálázhatóság

Ha a szerverek az átmeneti tárolással sem tudják kiszolgálni a megnövekedett látogatószámot, a rendszert bővíteni kell. Az olyan rendszert, ami könnyen és alacsony költségen bővíthető, jól skálázhatónak nevezzük. Figyeljünk oda, mert menedzser kollégáink nem mindig tudják mit jelent ez a kifejezés, és gyakran összekeverik a skálázhatóságot *(scalability)* a teljesítménnyel *(performance)*. A skálázhatóság nem azt jelenti, hogy a rendszerünk gyorsabb lesz, csak azt, hogy több felhasználót tud lassulás nélkül kiszolgálni.

A .Net környezetben épített rendszerek, ha jól vannak kitalálva, igen jól skálázhatók, nem kell mást tenni, mint egy-egy új szervert vásárolni, telepíteni a szoftvert, csatlakoztatni a közös adatbázishoz és az új szerver máris futásra kész. A meglévő szerverekhez pedig nem is kell hozzányúlni. Az ilyen skálázható megoldást hálózati terheléselosztásos megoldásnak nevezzük, hiszen a hálózaton keresztül bejövő lekérdezések sokasága, a terhelés, megoszlik több szerver között.

A több webszerverrel működő rendszereket web farmnak is szokás hívni. A Windows 2000 Advanced Serveren futó portál akár 24 szerverig is kibővíthető, így a legnagyobb országos Internetes vagy Intranetes portálok terhelésének többszörösét is ki tudja szolgálni, persze ehhez a mögé rakott adatbázist is tuningolni kell némiképpen. A Windows 2000 Advanced Server operációs rendszernek köszönhetően a terheléselosztás *(NLBS – Network Load Balancing System)* külön költséges hardver megvásárlása nélkül megoldható.

Nagy rendelkezésre állás

Az NLBS terheléselosztás másik nagy előnye, hogy a több szerver megléte redundanciát visz a rendszerbe. Ha a több webszerver közül az egyik meghibásodik, a többi még vígan működik, és átveszi a kiesett gép terhelését. Az NLBS megoldással tehát több legyet ütünk egy csapásra: skálázható lesz a rendszerünk, megnő a rendelkezésre állás, és egyszerűbb lesz az üzemeltetés, hiszen a karbantartást, ha egyszerre csak egy szervert állítunk le, több ideig lehet végezni, anélkül, hogy a szolgáltatás megállna.

Az SPF *(Single Point of Failure)* egy ilyen webfarmon általában az adatbázisszerver – amit a portálszoftver használ –, illetve maguk a hálózati eszközök, switchek, routerek és tűzfalak. A megfelelő hardver kiválasztásával a Microsoft SQL 2000 Server is megduplázható, de ez komoly költségnövekedéssel jár. Hiába redundáns a portálunk, ha a mögötte lévő rendszerek állandóan meghalnak.

Tartalommenedzsment – CMS

A portál oldalain megjelenő információk egy része valamilyen alkalmazásból származik, de nagy többségük egyszerű szöveges vagy képi információ, főleg Internetportálok esetében hétköznapi webes tartalom. A portálokon szereplő hírek, cikkek, leírások vagy egyéb képekkel illusztrált szöveges írásművek összefoglaló neve: tartalom. A portál tartalmát előállíthatja egy szerkesztőség vagy a szervezet különféle területein dolgozó alkalmazottak saját maguk. Kívánatos cél, hogy a webes tartalmat az tegye föl a portálra, aki azt előállítja. Ha PR- vagy marketingszöveg, a kommunikációs osztály, ha műszaki leírás, a support, ha szerződésminta akkor a jogászok. A portálon futó tartalommenedzsment *(angolul Content Management System, CMS)* alkalmazás segítségével a tartalmat előállítók önállóan is könnyedén készíthetnek és publikálhatnak tartalmi elemeket az Interneten, Intraneten vagy Extraneten.

Ha az oldalunkat percenként több ezren nézik, egyperces átmeneti tárolás is több ezer adatbázis-lekérdezéstől mentesíti a szerverrendszert.

A jó CMS rendszer garantálja, hogy a tartalmi elemek mindig a megfelelő helyen, a megfelelő időben és megfelelő kinézettel jelenjenek meg, illetve igény szerint gondoskodik a megjelenő tartalom megjelenés előtti szerkesztői, vezetői jóváhagyásáról. A CMS minden modern portálszoftver része kell legyen, sőt gyakori, hogy vállalati portál projektek CMS bevezetésből „nőnek ki".

Kereső

Minden portál legfontosabb feladata az, hogy a felhasználót a lehető leggyorsabban és legegyszerűbben információval lássa el. Mivel a legtöbb portálon naponta több tucat hír, cikk, dokumentum vagy egyéb írásmű keletkezik *(nem beszélve az alkalmazásokban tároltakról)*, az információtengerben való keresésre szoftvermegoldást kell alkalmazni. Fontos, hogy portálunk képes legyen a tartalomban szabad szöveges kereséseket futtatni. Mivel a keresési igények igen változatosak és kiterjedhetnek más rendszerekre *(fájlszerverek, levelezőszerverek, dokumentumkezelő szoftverek, más alkalmazások, az Internet)* is, olyan portálmegoldást érdemes választani, amely rendelkezik a megfelelő interfésszel, API-val, hogy a nekünk és menedzsmentünknek tetsző keresőmegoldást ki tudjuk fejleszteni, anélkül, hogy az egész keresőt újra kellene írni, vagy a beépített keresőt kellene úgy elfogadnunk, ahogy azt kapjuk.

Csoportmunka támogatás

A legtöbb cég vagy szervezet munkatársai csoportokban dolgoznak, így a csoport hatékonysága nemcsak attól függ, hogy abban az egyes munkatársak milyen hatékonyan dolgoznak, hanem attól is, milyen hatékonyan tudnak összedolgozni. A portál erre is adhat megoldást, ha olyan alkalmazásokat telepítünk rá, amelyek támogatják a csoportmunkát.

Az ilyen szoftvereket csoportmunka-támogató, angolul groupware vagy collaboration néven emlegetik. Kétféle megoldás létezik, az egyik, amikor egyszerű funkciókat a portálon mini-alkalmazásként valósítanak meg, a másik, amikor egy csoportmunka-támogató alkalmazást csatolnak a portálhoz. Az előbbire példa egy csapat-naptár, amiben a csapat tagjai látják egymás bejegyzéseit, az utóbbira példa a Microsoft Exchange Server csatolása oly módon, hogy például a felhasználó pos-

taládája vagy Microsoft Outlook naptára is a portálon jelenik meg.
A .Net platformon futó portálmegoldás bármilyen Microsoft szoftverhez csatlakoztatható, sőt webszolgáltatások *(webservice)* használatával akár más platformon futó rendszereket is csatolhatunk.

Mini-alkalmazások

Minden portálban szükségesek kicsi, egyszerű, de hasznos miniprogramok, mini-alkalmazások, amelyek a portál felhasználói felületén található dobozkák mögötti üzleti logikát adják. A szavazógép, fórum, linkgyűjtemény, hírleválogató vagy hasonló mini-alkalmazás megvalósítására minden gyártónak van keretrendszere.
Fontos szempont, hogy nekünk minél kevesebb dologra kelljen figyelnünk, mert annál kevésbé lesz „bugos" amit írunk. A gyakorlat azt mutatja, hogy a teljes jogú fejlesztőnyelvek, mint a Visual Basic, a C# vagy a C++ hatékonyabban alkalmazható több rétegű fejlesztéshez, mint a különféle scriptnyelvek, például az ASP. Lényeges, hogy kódolás közben ne nekünk kelljen jogosultságkezelést, cache-elést, vagy NLBS esetén sessionkezelést megvalósítanunk. A jó portál ezeket elrejti előlünk, ne is lássuk!

Többrétegű fejlesztés

Az egyszerű szoftverek egyetlen egységet alkotnak, nem modulokból állnak össze. Az ilyen szoftvereket monolitikusnak mondjuk. Előnyük, hogy egyszerűek, de a folyamatos továbbfejlesztésük nehézkes, főleg ha egy csapat dolgozik rajta. A komoly, nagy teljesítményű, skálázható és nagy rendelkezésre állású szoftverek gyártása más hozzáállást igényel. A komoly szoftverek modulokból, komponensekből állnak össze. A komponensek jól elkülöníthető feladatokat látnak el, egymás nélkül is újra felhasználhatók, külön-külön is továbbfejleszthetők vagy javíthatók, az egyes komponensek fejlesztési feladatai a fejlesztőcsapatban szétoszthatók. A portálokra pakolható dobozkák, mini-alkalmazások pedig általában több gyártótól származnak. A hálózaton keresztül működő üzleti szoftverek, így a portálok nagy része is, három alapvető logikai rétegből állnak: az adatkezelésből, az üzleti logikából és a prezentációból. Az alsó réteg csak az adatokkal foglalkozik, azok biztonságos tárolását és előhívását valósítja meg. A második, középső réteg a szoftver üzleti logikáját adja, azaz azokat az üzleti szabályokat valósítja meg, amelyek a szoftver lényegét alkotják. A harmadik réteg a szoftvert prezentálja, azaz megjeleníti a felhasználónak, biztosítja a felhasználói beavatkozás lehetőségét, megvalósítja a felhasználói élményt. A három réteg tovább bontható, ezért az ilyen fejlesztést három rétegű *(three tier)* vagy több rétegű *(n tier)* fejlesztésnek szokás nevezni.
Az olyan portálmegoldás, amely a prezentációs réteget és az üzleti logikát megfelelően elválasztja, azaz helyesen követi e többrétegű fejlesztés irányelveit, megkönnyíti a felhasználói felület fejlesztését. Így lesz könnyen változtatható, a cég arculatára szabható és egységes a portál felhasználói felülete. Az egységesség miatt nem kell az embereket többször oktatni, lecsökken a támogatás költsége.

A többrétegű fejlesztés nagy fegyelmet igényel, mert csak abban az esetben hajt hasznot, ha annak szabályait pontosan betartják. A rosszul megtervezett többrétegű alkalmazás továbbfejlesztése semmivel sem könnyebb, mint egy monolitikus alkalmazásé. A jó portálmotor tehát úgy valósítja meg az alkalmazásintegrációt, hogy keretet ad szabványos prezentációs és üzleti logika réteg építéséhez a fejlesztők kezébe, ezáltal elősegíti a többrétegű fejlesztés szabályainak betartását.

Internet portál – egy tartalom, két prezentáció

Fejlesztők támogatása

Mint minden IT beruházás esetén, portál építésnél is nagy hangsúlyt fektetnek a szervezetek vezetői az időre. Ki ne hallott volna „tegnapra" megrendelt Intranet, Internet vagy Extranet portálokról? Mivel az idő *(és általában a pénz is)* kritikus tényező, minden lehetőséget meg kell adni a fejlesztőknek, hogy munkájukat hatékonyan, kényelmesen és – nem utolsó sorban – jó minőségben végezhessék. A Microsoft világban az egyes számú eszköz a Visual Studio .NET, amely – mint minden Microsoft termék – testreszabható, és kiegészíthető úgynevezett add-in modulokkal. Ha tehetjük, használjunk olyan portálszoftvert, amihez lehet olyan kiegészítőket kapni, amelyek segítik a szoftver konfigurálását, a dobozkák mögött rejlő kód fejlesztését, a tesztelést és a dokumentáció készítését!

Szabványok

Nem elég azonban a szabályokat betartani, és szépen megtervezett több rétegű alkalmazásokat fejleszteni. Fontos, hogy minél több olyan szabványt alkalmazzunk, amelyeket más fejlesztők is ismernek és használnak. Ha így teszünk, sokkal könnyebb lesz az alkalmazások integrációja, lecsökken az oktatásra szánt idő és költség, könnyebb lesz új munkaerőt bevonni a fejlesztésbe. A ma kapható portálcsomagok élvonalába tartozó termékek mind az XML-szabványra épülnek, ezáltal bármilyen hardver- és szoftverkörnyezetben futó alkalmazás könnyen és hatékonyan integrálható segítségükkel.
A webszolgáltatások is szabványokra épülnek, de sajnos a piacon uralkodó üzleti harc miatt nem teljesen kompatibilisek egymással, noha állítólag mind a szabványokat követi. Szerencsére egyre több szoftvergyártó jön rá, hogy az inkompatibilitás csak árt az üzletnek, így az SAP például a JAVA világ megoldásait és a .Net-et is egyformán támogatja. Az ODBC és egyéb szabványokról azt hiszem nem kell sokat mesélni, de érdemes megemlíteni a Microsoft Host Integration Servert, ami sok különféle nem Microsoft szoftverhez ad hozzáférést, illetve a Microsoft BizTalk Servert, ami pedig az elektronikus kereskedelmi megoldások építésénél vehet le sok terhet a vállunkról.

Design

Internetportáloknál gyakran felmerül az igény, hogy a portál ne szokványos, dobozkákból álló felhasználói felület legyen, hanem kreatív kommunikációs csatorna, amelynek csak a fantá-

zia és a sávszélesség szab határt. A jó portálmotor erre is lehetőséget ad, támogatja a grafikailag összetettebb, de mégis böngészőfüggetlen webdesignt. Ha a rendszerünk a Macromedia Flasht is megjeleníti, sőt dinamikus tartalommal is el tudja azt látni, teljes a designerek szabadsága. *(Azért gondoljunk a modernesekre is, és kíméljük meg látogatóinkat a másfél órás intróktól.)*

Alternatív megjelenés – mobileszközök

Ha a portálmotor a prezentációs réteget és az üzleti logikát megfelelően elválasztja, a portálon működő alkalmazások és a portálon elérhető információk alternatív böngészőkben, például mobileszközökön is egyszerűen megjeleníthetők. Egy tartalmi elem vagy egy szolgáltatás megjelenését több böngészőre vagy eszközre lehet optimalizálni. Amíg egy bizonyos megjelenítő gazdagon díszített DHTML vagy XHTML kimenetet állít elő asztali számítógépen futó, újabb verziójú böngészők számára, egy másik megjelenítő előállíthat egyszerű HTML, WAP vagy XML kimenetet is, amit mobileszközök, kézi számítógépek, mobiltelefonok vagy régebbi böngészők is kiválóan meg tudnak jeleníteni, akár igen lassú Internet kapcsolaton keresztül is.

Közgáz – ROI

A portálprojektbe fektetett idő és pénz sokféleképpen térülhet meg. A dolgozók hatékonysága megnő, információk után való szaladgálás helyett valami értelmeset és hasznosat tesznek. Jó környezetben jobb dolgozni, és jó kedvvel minden munkatársunk hatékonyabb. Főnökeink, ügyfeleink vagy beszállítóink jólinformáltak lesznek, könnyebben és gyorsabban hoznak olyan döntéseket, amelyek nekünk hasznot hajtanak. A fenti előnyöket nehezebb számokban kifejezni, de egyes kutatások rámutatnak, hogy az egységes, központosított infrastruktúrán megvalósított Intranet-, Extranet- és Internetportálok megtérülése igen gyors. Ha pedig több kicsi portálra van szükségünk, akár szervezeti egységek, akár témák szerint, mindenképpen válasszunk olyan portált, ahol az információk több portálon is megjeleníthetők. Figyeljük meg, hogy a közös infrastruktúra megléte kihat a TCO-ra is.

Közgáz – TCO

A megfelelő felügyeleti eszközök, logolvasók, adatbáziskarbantartó scriptek, öndokumentáló karbantartási eljárások, távfelügyelet, automatikus felügyelet és öndiagnosztika mind csökkentik a portál birtoklásának összköltségét. Ha például az adminisztrációs felület szintén egy portál, azaz webes alkalmazás, nem kell klienseket telepíteni. Ez az intranet egyik legnagyobb, bár nem egyetlen előnye. Az NLBS lehetőséget ad a gépek egyenkénti, teljes leállás nélküli frissítésére, így nincs munkaidőkiesés. Ez az úgynevezett rolling upgrade lehetősége a 24 órában az Internetre kötött rendszerek esetében még a legforróbb tűzfal mellett is kritikus a biztonság szempontjából.

Zárszó

Természetesen ebbe a cikkbe sem fér bele minden, amit a portálokról tudni érdemes. Az alkalmazásszerverek, a tranzakciókezelés, a biztonság, a többnyelvűség vagy az off-line elérhetőség kérdései szintén igen fontosak. Kérek mindenkit, hogy véleményét juttassa el e-mailben az írónak, vagy a szerkesztőségnek, ezzel is segítve a sorozat következő témájának kiválasztását. Végezetül álljon itt egy checklist arról, mit tud a jó portál:

- XML-, XSLT-alapú prezentáció
- Web Service, SOAP támogatás
- Intranet, Internet, Extranet egy rendszerben
- Skálázhatóság és nagy rendelkezésre állás
- Fejlesztő, tesztelő és dokumentáló eszközök
- Tranzakciókezelés lehetősége
- Beépített cache, és cache API
- Egy jelszó, SSO
- Kereső, Indexelő megoldás, kereső API
- CMS, sablonok
- Biztonság, titkosítás, SSL *(https://)*
- Mobileszközök támogatása
- Többnyelvűség, Magyar GUI
- Hazai support.

Ha mindezeket figyelembe vesszük, és az üzleti célokra is odafigyelünk, az eredmény egy dinamikus portál, amely nagy rendelkezésre állás mellett képes kiszolgálni az üzleti igényeket és az egyre növekvő terhelést, jöjjön az házon belülről vagy házon kívülről.
A felhasználói élményt az Egy Jelszó és a mini-alkalmazások adják, a rendszerintegrátorok és fejlesztők munkáját pedig a szabványokra épülő többrétegű fejlesztőkörnyezet és a fejlesztést segítő eszközök biztosítják. Sok sikert!

Bíró Tamás
bt@sensenet.hu
MCSD

Az Active Directory programozott elérése

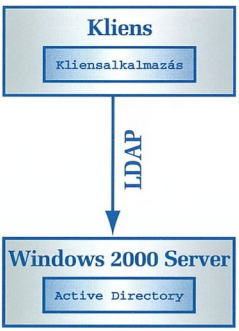
I. rész

Sokszor esett már szó az Active Directory eléréséről, bővüléséről, most azonban a saját kódból történő elérést szeretném bemutatni, illetve illusztrálni néhány példán keresztül. Úgy gondolom, ezek az ismeretek mind fontosak lehetnek egy AD-integrált alkalmazás fejlesztésekor.

Első lépésként néhány technológiát szeretnék bemutatni, amelyek segítségével hozzáférhetünk az Active Directory tartalmához.

LDAP *(Lightweight Directory Access Protocol)*

Az LDAP alacsony szintű kódok segítségével teremt közvetlen kapcsolatot a kliens és az Active Directory között. Legbelül a többi, közkedvelt kapcsolódási technológia, az ADSI és a COM is ezt használja. Természetesen ez utóbbiak lassabbak, mivel magasabb szintű szolgáltatást nyújtanak.

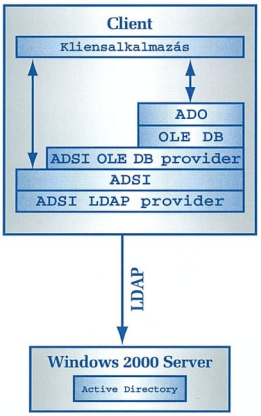


Az LDAP struktúrája

Az LDAP segítségével mindemellett nemcsak az AD kezelhető, hanem bármely címtárszolgáltatás is.

ADSI *(Active Directory Service Interface)*

Az ADSI egy COM-alapú, LDAP-on keresztül kommunikáló interfész, amely minden olyan esetben használható, ahol a COM-os megoldások megállják helyüket. Kevesebb kódot igényel, mint az LDAP, viszont működése lassabb is. Mind kliens-, mind szerveroldalon telepíteni kell *(a Windows2000-rel automatikusan jár)*.



Az ADSI struktúrája

A kliens vagy közvetlenül az ADSI-val kommunikál, vagy ADO-n keresztül éri el azt. Az ADSI alatt még egy réteg, az LDAP provider helyezkedik el, a kommunikáció közvetlenül ezen keresztül zajlik.

CDO *(Collaboration Data Objects)*

A CDO szintén COM-alapú technológia, viszont kevésbé gazdag, mint az ADSI. Kliensoldalra nem lehet installálni, így több feladat hárul a kiszolgálóra.

Jogosan merül fel a kérdés, hogy a fenti megoldások közül mikor melyiket érdemes alkalmazni. Hasonlóságuk alapján az ADSI-t és a CDO-t érdemes összehasonlítani, hiszen alacsonyszintű, nagysebességű megoldásával a „csupasz" LDAP külön kategóriát képez.
Egy igen jelentős szempont, hogy kliens- vagy szerveroldali alkalmazást írunk-e. Ha a kiszolgáló feladatait minimálisra szeretnénk csökkenteni, az ADSI-t célszerű használni, míg ellenkező esetben CDO-t. CDO ajánlott akkor is, ha az alkalmazásunk gyakran hoz létre vagy módosít felhasználókat, Exchange mailboxokat és mozgat információkat az AD és a WSS mappák között. Ha általános AD-kezelési feladatok elvégzését szeretnénk egyszerűen megvalósítani, az ADSI használata javasolt.

Az LDAP elérési út

Bármelyik megoldást választjuk is, mélyen legalul LDAP-ot használunk, így nélkülözhetetlen az LDAP formátumú elérési utak ismerete: az ADSI az objektumokhoz történő hozzáféréshez *(binding)*, a CDO az adatforrás megnyitásához használja őket.

Az LDAP név alakja a következő: `LDAP://server/DN`. A `server` egyértelműen a kiszolgáló neve, a DN pedig az Active Directory-beli objektum neve *(distinguished name)*. Ha DN-t nem adunk meg, a beléptetés megtörténik az aktuális felhasználó nevében, és az ő jogaival látjuk az Active Directory legfelső szintjét.

A `DN` három részből áll:

1. `CN` *(common name)*: az objektum neve.
2. `OU` *(organizational unit)*: – az a szervezeti egység, ahova az objektum tartozik
3. `DC` *(domain controller)*: a domain controller nevének darabjai *(például `DC=msdn,DC=microsoft,DC=com`)*

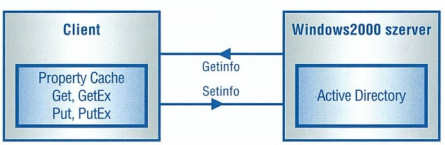
A DN a fenti elemek tetszőleges kombinációját tartalmazhatja, azonban a sorrend kötött *(a fenti felsorolás szerint)*. Így például egy LDAP név lehet a következő:

```
LDAP://mydomain.encegem.hu/CN=Agnes,CN=Molnar,
↳ OU=Developers,DC=encegem,DC=hu
```

A Property cache

Mivel az alkalmazásunk igen kevés esetben futhat magán a szerveren *(a domain controlleren)*, a hálózat terheltségének csökkentése érdekében a kliens és szerver közötti kommunikációt minimalizálni kell *(és az alkalmazásunk is gyorsabb lesz így)*.

A kliens gépen egy úgynevezett property cache jön létre, amelybe a szerverről érkezett adatokat elhelyezzük. Ha az AD-ból valamilyen új adatot kapunk, elhelyezésre kerül ebben a



cache-ben, és legközelebb közvetlenül innen érhetjük el.

Az AD adatait az ábrán látható metódusokkal kezelhetjük. Első alkalommal feltöltésre kerül a property cache. Ez egyrészt a Get vagy GetEx metódus meghívásával, implicit történhet, melyek – mivel az adat nem található a cache-ben – az AD-hez fordulnak; másrészt explicit módon, a GetInfo meghívásával. Ha a cache-t feltöltjük, a Get és GetEx metódusok előbb onnan próbálják kinyerni a kívánt adatot, s csak sikertelenség esetén fordulnak a szerverhez. A GetInfo mindig a szerverhez nyúl az adatokért.

A jellemzők cache-beli értékeinek beállításához a Put és PutEx metódusok használhatók. Ezek mindig a cache-be írnak, akkor is, ha az adott jellemző korábban még nem szerepelt ott. Későbbi Get eredményeképp az AD-ből bekerülő adatok felülírják a Put által oda helyezetteket. Ezért legalább egy Get metódusnak meg kell előznie a Put-okat.

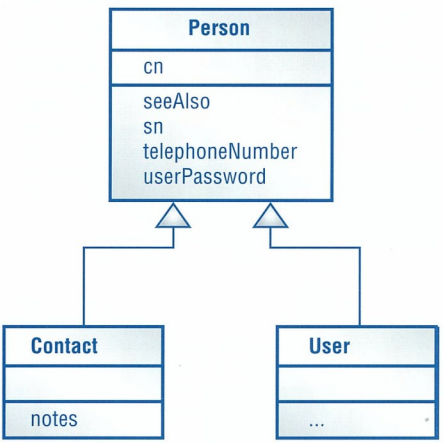
Az AD-be történő íráshoz a SetInfo metódus használható. Ez a cache-ben található összes adatot feltölti az Active Directory-ba. Ha valamely érték érvénytelen, akkor hibaüzenetet kapunk.

A Get és GetEx, illetve a Put és PutEx metódusok között az az alapvető különbség, hogy az előbbiek egyetlen érték kezelésére használandók, míg az utóbbiak többértékű jellemzők olvasására illetve írására szolgálnak.

Az Active Directory felhasználókat reprezentáló elemei

Az AD-ben kétféle objektum reprezentálhat személyeket: a Contact és a User. Mindkét típus képes e-mailek fogadására, de csak a User-hez rendelhető Exchange mailbox és security környezet.

A személyeket megtestesítő objektumok a Person sémaosztályon alapulnak. Egy Contact-objektum egy olyan AD-bejegyzés, amely alapvető információkat tartalmaz egy személyről: név, cím, telefonszám, stb. A User-objektum „security principal": be tud lépni a hálózatra, és különböző jogosultságokkal rendelkezik a hálózati erőforrásokhoz *(nyomtatás, fax, stb.)*. A Contact-nak nincsenek ilyen jogosultságai, csupán adataival szerepel az Active Directory nyilvántartásában *(ennek célja például az, hogy könnyedén hozzáférhessünk, levelet küldhessünk neki stb., viszont az illető személy ne garázdálkodhasson a vállalati hálózaton)*.



Contact és User elemek az Active Directoryban

A fenti ábra tehát a Person osztályból történő leszármazást mutatja. *(Annyi apró „trükk" került az ábrába, hogy az objektumosztályokat szemléltető téglalapok középső részén a kötelező, míg alsó harmadában az opcionális attribútumokat tüntettem fel.)*

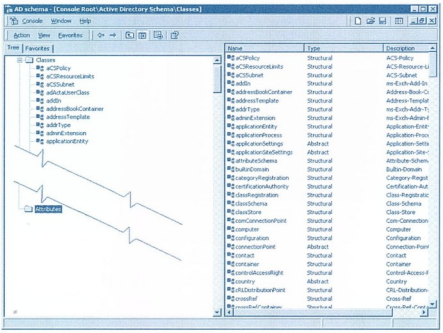
Az Active Directory séma

A séma az Active Directory objektumait írja le. Mivel a sémadefiníció maga is az Active Directoryban, objektumok formájában tárolódik, ugyanúgy adminisztrálható, mint bármely „hétköznapi" Active Directory objektum – természetesen a megfelelő jogosultságok birtokában.

A séma kétféle objektumot tartalmaz: osztály- és attribútum-leírásokat. Ezeket összefoglaló néven általában sémaobjektumoknak vagy metaadatoknak nevezzük.

Az osztályleíró objektumok az Active Directory objektumainak lehetséges típusait határozzák meg. Sablonként funkcionálnak, melyeket új objektum létrehozásakor használhatunk, és annak

leírására szolgálnak. Azokat az attribútumokat tárolják, amelyek az adott típusú objektumok jellemzőiként funkcionál*(hat)*nak.
Minden AD-beli objektum valamely AD-osztály példánya.
Az attribútumleíró sémaobjektumok az Active Directory-beli jellemzőket, attribútumokat írják le. Minden attribútumot egyszer kell definiálni, később annyi osztályban használhatjuk, ahányban csak jólesik.



Az Active Directory séma – osztályok és attribútumok leírása

A Windows2000 Serverrel szállított alap-séma természetesen kibővíthető. Fontos tudni, hogy a sémába újonnan felvett elemek soha többé nem törölhetők onnan, típusuk, nevük nem változtatható, így rendkívüli körültekintést igényel a séma bővítése *(mi azonban természetesen megpróbálkozunk vele a későbbiekben ☺)*.

Az alábbi táblázat az osztályleíró sémaobjektum kulcsfontosságú attribútumainak listáját tartalmazza:

LDAPDisplayName	Leírás
cn *(Common Name)*	Minden AD objektum rendelkezik egy őt azonosító névvel, amely az ő RDN-je. A **`classSchema`** megnevezésére a cn szolgál. A cn-nek egyedinek kell lennie a schema mappán belül.
LDAPDisplayName	Az LDAP kliensek által használt név, ennek segítségével hivatkozhatunk az osztályra. A séma mappán belül egyedinek kell lennie.
schemaIDGUID	Nyolc karakterből álló *(octet)* string. Egyedi, az osztály azonosítására szolgál. A bejegyzésekhez történő hozzáférésre használható. Ha új **`classSchema`**-t hozunk létre, az AD automatikusan legenerálja, amennyiben mi nem készítünk saját GUID-ot. Ajánlott saját azonosítót létrehozni minden osztályhoz.
adminDisplayName	Az adminisztrációs eszközökben megjelenő név. Ha nem specifikáljuk, akkor létrehozunk egy osztályt, a rendszer a CN-t használja. *(Csak akkor használja a rendszer, ha a **`classDisplayName`** bejegyzés üres.)*
mustContain, systemMustContain	Ezek a többértékű jellemzők azt specifikálják, hogy mely attribútumokat kötelező beállítani az adott osztály példányain. Kötelező attribútumok, amelyeket már az objektum létrehozásakor be kell állítani, és a későbbiek során sem törölhetőek, sőt nem is változtathatók.
mayContain, systemMayContain	Ezek a többértékű jellemzők azt specifikálják, hogy mely attribútumokat állíthatjuk be az adott osztály példányain. Ezek opcionális attribútumok, megadásuk nem kötelező. 1. és 2. kategóriás **`classSchema`** objektumból egyaránt válogathatunk ide. Mielőtt az osztálydefinícióból törlünk egy attribútumot, az összes példányból ki kell az törölni. Az osztály létrehozása után a **`systemMayContain`** jellemző nem módosítható. Az opcionális attribútumok teljes halmaza a **`systemMayContain`** és a **`mayContain`** uniója, ezen és az összes örökölt osztályon.
possSuperiors, systemPossSuperiors	Azokat a strukturális osztályokat specifikálja, amelyek valódi szülei lehetnek ezen osztály példányainak. A teljes készlet e két jellemző, és az örökölt strukturális vagy absztrakt osztályok uniója. A jellemzők értékei az örökölt és kisegítő osztályokra nem érvényesek. A possSuperiors bármikor változtatható, törölhető, a systemPossSuperiors viszont az osztály létrehozásakor állítható be, később nem módosítható, és nem távolítható el.
objectClassCategory	Integer érték, mely az osztály kategóriáját mutatja.
subClassOf	Az osztály közvetlen ősosztályának OID-je. A hivatkozott ősosztálynak léteznie kell, különben meghiúsul az osztály hozzáadása a sémához.

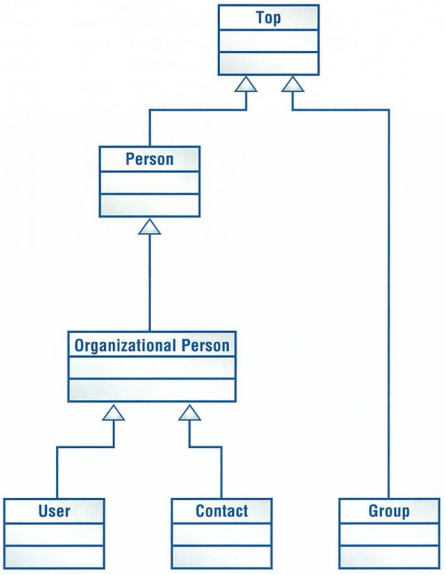
Öröklődés

Természetesen az Active Directoryban is definiált az öröklődés fogalma. Minden objektumosztály a speciális, legfelső top osztály leszármazottja. A top kivételével minden objektumosztály leszármazottja valamely másik osztálynak. *(Tehát az egész hierarchia egy fába rendezhető, amelynek gyökéreleme a top osztály.)* Néhány attribútum értéke az ősosztályból öröklődik, ilyenek például:

- **Lehetséges attribútumok.** Egy **`classSchema`** objektum **`mustContain, mayContain, systemMust-`**

`Contain` és `systemMayContain` attribútumértékei határozzák meg azon attribútumok teljes listáját, amelyek egy osztály példányán megjelenhetnek. Minden egyes objektumosztály örökli a fenti attribútumok értékeit az ősosztályától, és minden értéket erre állít be.

- **Lehetséges ősök a directory hierarchiában.** A `possSuperiors` és `systemPossSuperiors` attribútumok definiálják azon objektumosztályok teljes listáját, amelyek tartalmazhatnak objektumosztály-példányokat. Minden objektumosztály az ősosztályától örökli ezeket az értékeket.



Öröklődés az Active Directoryban

Ennyi bevezetés után térjünk végre a lényegre, hiszen programozásról eddig szó nem volt. A nyitott kérdéseket folyamatosan, a programozási problémák ismertetése és megoldása közben zárom majd le.

Néhány egyszerű programozási feladat

Objektum létrehozásához első lépésként aktív kapcsolatot kell kiépítenünk azzal a mappával, ahova létre akarjuk hozni az új objektumot *(felhasználót, csoportot, stb.)*, majd meghívjuk az `IADsContainer` interfész `Create` metódusát. Ez szolgál objektumok létrehozására, és két paramétere van:

- A létrehozandó objektum sémaosztályának neve *(pl. group, user, stb.)*
- Az új objektum relatív DN-je *(pl. `CN=Agnes`)*

Új csoport létrehozására szolgál az alábbi kódrészlet:

```
Dim ctnr As IADsContainer
Dim grp as IADsGroup

Set cntr =
➥ GetObject(„LDAP://cyberserver/ou=recipiens,
➥ dc=domain,dc=com")
Set grp = cntr.Create(„group", „cn=ADhackerek")
grp.put(„description", „Ez az uj csoport")

grp.SetInfo
```

Hasonló módon hozhatunk létre például felhasználókat is, ez esetben az `IADsGroup` helyett az `IADsUser`, általános Active Directory-objektum esetében az `IADs` interfész használandó.

Objektumok törlése is hasonló, egyszerű módon végezhető, a `.Delete` metódus segítségével. Például felhasználót így törölhetünk:

```
Set cntr =
➥ GetObject(„LDAP://cyberserver/ou=recipiens,
➥ dc=domain,dc=com")
cntr.Delete(„user", „cn=Agnes Molnar")
```

Ezekből az egyszerű kódokból jól látható, hogy az Active Directory „bütykölése" nem nagy ördöngösség. De vajon bonyolultabb, összetettebb feladatokra igaz-e ez? Például a sémamódosítás is ilyen egyszerűen végrehajtható? Ehhez hasonló kérdésekre keressük a választ a következő hónapokban. Segítségül ismét egy összetettebb példát hívok, hogy ezáltal szemléletesebben és érthetőbben vezethessek be mindenkit az AD programozásának rejtelmeibe.

Molnár Ágnes
agi@harpia.homeip.net

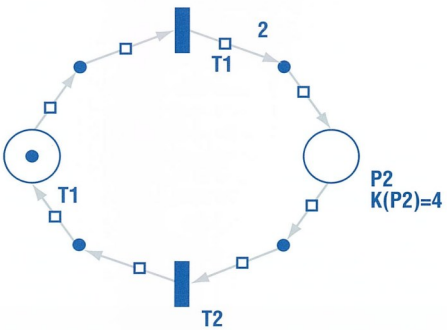
Formális módszerek az informatikában [2]

A Petri-hálók további alkalmazásai

A sorozat első részeként áttekintettünk néhány alapfogalmat az informatikában használatos formális módszerekkel kapcsolatban, majd áttértünk a Petri-hálókra. A topológia bemutatása után most mélyebb részleteket szeretnék bemutatni.

A kapacitáskorlát kiküszöbölése

Lássunk egy nagyon egyszerű példát. Az alábbi ábrán egy kapacitáskorlátos Petri-háló látható, melynek működése könnyen leírható: a kiindulási helyzetben egyetlen tokenünk van, a P1 helyen. A T1 tranzakció tüzelési feltétele, hogy P1-en legyen token, ezúttal ez teljesül. A T2 tranzakció jelenleg nem engedélyezett, hiszen P2-ben nincs token.



Egyszerű kapacitáskorlátos Petri-háló

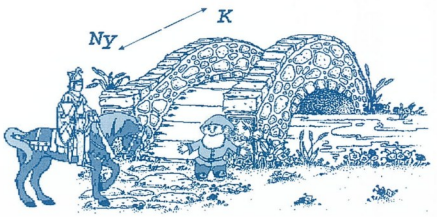
A T1 tranzakció tüzeléskor T2-be 2 tokent tesz, vagyis eggyel növeli a rendszerben keringő tokenek számát. T2 ezt a szumma tokenszámot nem változtatja, csupán elvesz P2-ből egyet, és azt P1-be helyezi át.

Jól látható, hogy T1 minden egyes tüzelése eggyel megnöveli a tokenek számát, vagyis nincs felső korlát arra, hogy a rendszerben mennyi token lesz, ha tetszőlegesen hosszú ideig magára hagyjuk futás közben. Ezáltal arra sem tudunk korlátot adni, hogy egy-egy állapotban hány tokenünk lesz. Ezt hivatott megoldani a kapacitáskorlát, amiről előző cikkemben már írtam. Ha bevezetünk egy K*(P2)*=4 kapacitáskorlátot, az azt jelenti, hogy a P2 helyen maximum 4 token lehet egyszerre, s ez a T1 tranzíció tüzeléséhez is egy újabb, korlátozó feltételt jelent *(ha P2-n már 4 token tartózkodik, akkor a T1 tüzelése nem engedélyezett, hiába van token P1-en).*

Ez a módszer megoldást jelent tehát bizonyos problémákra, sajnos azonban több analízis-módszer *(ezekről lásd később)* nem támogatja használatát, így amit nyertünk vele az egyik oldalon, azt megfizetjük a másikon. De mi furfangos és telhetetlen informatikusok lévén szeretjük megkeresni a kiskapukat. Így teszünk most is, vagyis funkcionalitásukban kapacitáskorlátos helyeket szeretnénk megvalósítani konkrét kapacitáskorlát nélkül.

Vajon hogyan lehetséges mindez?

Gondoljunk bele, mi történik a P2 hellyel, amikor elveszünk onnan, illetve teszünk oda tokent. Tegyük fel, hogy P2 egy kapacitáskorlát nélküli *(végtelen korlátos)* hely, és kezdetben nincs ott token *(ez a helyzet a fenti példában is).* Képzeljük el, hogy a P2 helyen *(a híd lábánál)* egy törpe ücsörög, akinek üres az erszénye. Ha arra jár egy lovag *(és mondjuk keleti irányba tart)*, egy garas vámot szed tőle, ha viszont ugyanez a lovag visszafelé jön, visszakapja a garasát. Így a törpénél átmenetileg lehetnek a garasok, és maximum annyi, ahány lovag kelet felé elhaladt.



A törpe és a lovag

Ha a törpe zsebébe k darab garas fér el, egyszerre k lovag lehet a hídtól keletre *(tegyük fel, hogy a világ azon felén eredetileg nem éltek lovagok).* Ebben az esetben csak akkor mehet át a hídon a következő lovag, amikor egy másik visszatér. *(Formálisan: a Lovag_átlép tranzíció csak akkor engedélyezett, ha a Kelet állapotban k-nál kevesebb token van.)*

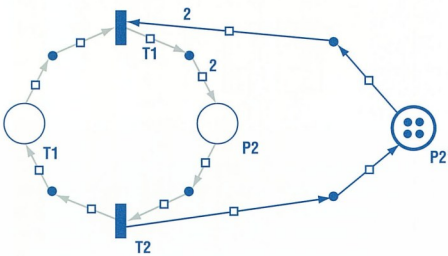
Igen ám, de ha mesebeli törpénk zsebe feneketlen, elvileg akárhány lovag lehet a keleti, és a nyugati oldalon egyaránt. Hogyan oldható meg ekkor, hogy a sok lovag ne árassza el a keleti oldalt? Ennek egyetlen módja van: kevés embert kell lovaggá ütni. Ha az Óperenciás tengeren túli király gondoskodik arról, hogy országában összesen k lovag legyen *(és semmiképpen sem több)*, akkor egészen biztos, hogy a keleti oldalon sohasem lesz k-nál több lovag. Ebben az esetben, ha keleten x számú lovag van, akkor nyugaton egészen pontosan k-x.

Mit jelent ez a Petri-hálók nyelvezetére lefordítva?

Ha azt akarjuk elérni, hogy a P2 *(kelet)* helyen maximum k=4 darab token legyen, be kell vezetnünk egy adminisztrációs helyet *(P2')*, ahol azt tartjuk számon, hogy a P2 helyre hány to-

ken fér még el *(nyugat)*. Így az adott helyen lévő tokenek számát m-mel jelölve mindig igaz lesz az alábbi összefüggés: m*(P2')*+m*(P2)*=k.

De hogyan felügyelhető, hogy ez minden esetben így legyen? Kezdetben a P2' helyen legyen k darab token, P2-n pedig 0. A P2-vel szomszédos tranzakciókhoz vegyünk fel új éleket az alábbiaknak megfelelően: ha a tranzíció a darab tokent vesz el a P2 helyről, akkor az új élen adjon a darabot a P2'-höz. Ha pedig b tokent ad P2-höz, akkor ugyanennyit vegyen el P2'-ből. A fenti példából kiinduló Petri-hálónk tehát így néz ki:



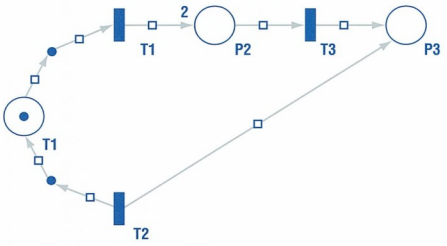
A módosított Petri-háló kapacitáskorlát nélkül

A módosított T1 tranzíció tehát nemcsak hozzáad P2-höz 2 db. tokent, hanem ugyanennyit el is vesz a P2' adminisztrációs helyről. Hasonlóan T2 elvesz 1 tokent P2-ből, és hozzáad egyet P2'-höz.

Szimuláció Petri-hálókkal

A Petri-háló tevékenységeit, akcióit olyan elemi *(atomi)* eseményekre bontjuk, amelyek tovább már nem oszthatóak. Esetünkben atomi eseménynek mondunk egy tranzíció tüzelését, így az összetett események a tüzelési szekvenciákat jelentik *(az egymás után végrehajtható tüzelések sorozatát)*. Az eseményeket a rendszerben szereplő állapotváltozókkal szimuláljuk.

Petri-hálók számítógépes szimulációjához több dolgot figyelembe kell venni. A nemdeterminisztikus viselkedés miatt az időzítéseket *(ál)*véletlen módon kell megoldani, tehát ha több tüzelhető tranzíció is van egy adott időpillanatban, véletlenszerűen válasszuk ki azt, amelyik a következő alkalommal valóban tüzelni is fog. Ilyenkor azonban fennáll annak a veszélye, hogy ha egy tranzíció tüzel, olyan tranzíciók is tiltottá válhatnak, amelyek az előző lépésben még engedélyezettek, illetve olyanok válhatnak engedélyezetté, amelyek korábban tiltottak voltak. Lássuk például az alábbi ábrát:



Példa Petri-háló szimulációjára

A fenti ábrán látható kezdőállapotban mind a T1, mind a T2 tranzíciók engedélyezettek. Ha a véletlenszerű döntés után a T1 tüzel elsőként, több változás is beáll a rendszerben: a token a P1 helyről átkerül a P2-be, ezáltal a T1 és T2 tranzíciók letiltódnak. Ugyanakkor a T3 engedélyezett lesz, hiszen bemenetén egyetlen hely, a P2 szerepel, és ott is csupán a tokennek kell állnia, ez a feltétel pedig ekkor teljesül.

Ha meg akarjuk tehát valósítani a Petri-hálók algoritmusát, figyelembe kell venni ezeket a letiltódásokat és engedélyeződéseket. Első megközelítésben tehát végigvizsgáljuk az összes tranzíciót, hogy teljesül-e rá a tüzelési feltétel: ha igen, hozzáadjuk az engedélyezettek listájához *(ha eddig még nem szerepel rajta)*, illetve ha nem, elvesszük a listából.

Ezáltal elérjük, hogy minden tüzelés után aktuális képünk van a tüzelhető tranzíciókról, azonban ez a megvalósítás meglehetősen lassú: minden tüzelés után minden tranzíciót végig kell vizsgálnunk. Természetesen ennél hatékonyabb megvalósítás is lehetséges, ha figyelembe veszünk néhány egyszerűsítő tényt. Jelölje a T tranzíció őseit *(bemeneti helyeit)* •T, utódait *(kimeneti helyeit)* pedig T•. Hasonlóan jelöljük a •P hely őseit és utódait P-vel és P• -tal. Vegyük észre, hogy egy T tranzíció tüzelésekor csupán a •T helyekről veszünk el tokent, így csak azt kell vizsgálni, hogy ezen helyek utódai *(azaz a (•T)• tranzíciók)* tüzelhetők-e a továbbiakban *(vagy letiltódtak)*. Hasonlóan a T tranzíció tüzelésekor csak a T utódaiba kerül token, így ezek módosíthatják a további tüzelhetőséget, azaz *(T•)•* vizsgálatával állapíthatjuk meg, hogy engedélyeződtek-e újabb tranzíciók.

A fenti példában tehát T1 tüzelésekor a *(•T1)={P1}* állapotból veszünk el tokent, azaz a *((•T1)•)=({P1}•)={T1,T2}* tranzíciókat kell csupán megvizsgálni (T3 nem tiltódhatott le akkor sem, ha engedélyezett volt, mert nem utódja T1 egyetlen ősének sem). Ekkor azt kapjuk, hogy mind T1, mind T2 tiltott állapotba került, ugyanis bemenetükön nincs meg a megfelelő számú token.

Lássuk, lett-e engedélyezett token. T1 utóda egyedül a *(T1•)={P2}* hely. P2 utód-tranzíciói: *((T1•)•)=({P2}•)={T3}*. T3 vizsgálatakor kiderül, hogy eddig ugyan tiltott volt, most azonban már van a bemenetén a szükséges számú *(1 db)* token, tehát engedélyezett lett. Ha lenne a hálóban több tranzíció, azok tüzelhetősége nem változna, hiszen T1 tüzelése csupán a közvetlen környezetét változtatja meg.

A tüzelhetőség vizsgálata elsősorban olyan esetekben jó, amikor arra vagyunk kíváncsiak, hogy a hálóban van-e holtpont *(deadlock)*. Ennek vizsgálata létfontosságú, ugyanis a nem holtpontmentes rendszerek működésének folytonossága nem biztosított, fennáll annak lehetősége, hogy olyan helyzetbe kerülne, ahonnan sehova sem tudnak továbblépni *(nincs egyetlen engedélyezett tranzíció sem)*. Ekkor a szimulált rendszer „befagy", többé nem működőképes. Ez pedig végzetes lehet.

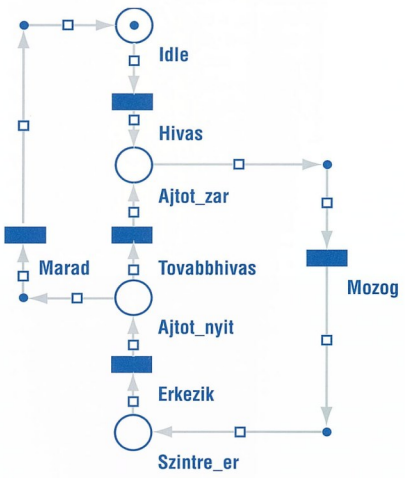
Tervezési példa Petri-hálókkal

Az elmúlt hónapban példaként szerepelt egy kétemeletes lift, amelynek akkor állapotautomatáját rajzoltuk fel. Most ennek Petri-hálós megvalósítását szeretném bemutatni.

Először is azonosítsuk a feladatunkban szereplő állapotokat *(ezek lesznek a Petri-háló helyei)* és állapotátmeneteit *(ezek lesznek majd a tranzíciók)*.

A lift minden egyes szinten lehet nyugalmi állapotban *(Idle)*, ha éppen sehova sem hívták, és utasa sincs. Ha éppen megérkezik egy emeletre, Erkezes állapotba lép, melyet az ajtó zárása előz meg, illetve az ajtó nyitása követ, természetesen a megfelelő emeleten.

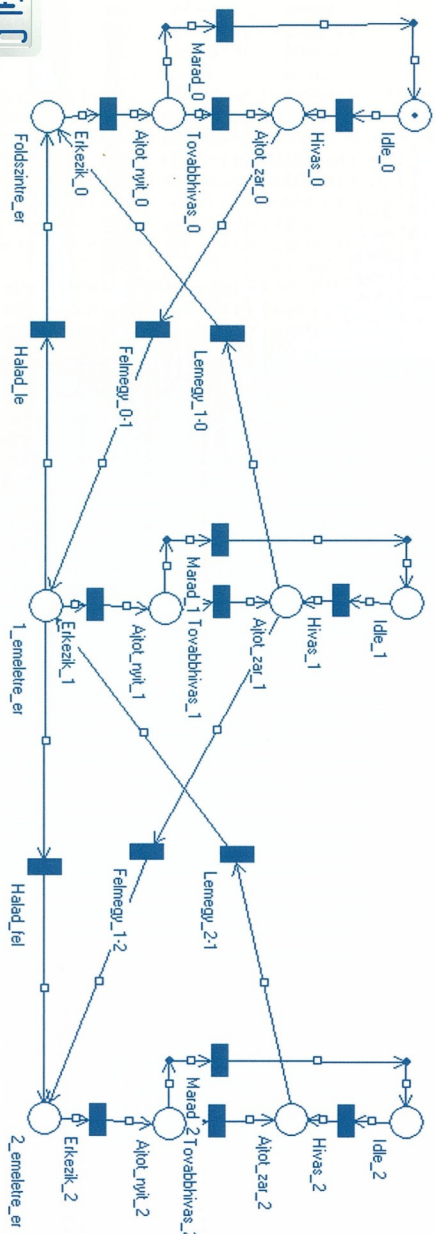
Az egyes állapotok közötti átmeneteket különböző események váltják ki. Nyugalomba akkor kerül a lift az adott szinten, ha teljesítette feladatát, és nincs több utasítás számára. Az ajtó akkor záródik be, ha elhívják a liftet: ekkor vagy nyugalomból „ébred fel", vagy korábbi mozgását folytatja a megfelelő irányba. Ajtaját akkor nyitja, ha megérkezett egy emeletre.
Az ajtó záródását és egy másik emeletre történő érkezést a lift adott irányú mozgása választja el egymástól. Ajtózáródás- és nyitódás nélkül csak akkor mozoghat a lift, ha áthalad egy emeleten. Esetünkben ezt egyedül az első szinten teheti meg.
Az alábbi ábra egyetlen szint Petri-hálóját mutatja, ahol a többi szinten történő mozgást és egyéb tevékenységeket a Mozog tranzíció hivatott helyettesíteni, abból a megfontolásból, hogy ha állunk egy emeleten és várjuk a liftet, lényegtelen, hogy az merre kószál az épületben, csak az számít, hogy éppen aktuálisan számunkra nem érhető el. *(Természetesen itt attól a lényeges tényezőtől eltekintünk, hogy esetleg sietünk valahova, így kardinális jelentőségű lehet az az információ is, hogy a lift éppen melyik állomáson tartózkodik.)* További feltételezésünk az egyszerűség érdekében, hogy a lift soha nem hibásodhat meg. Ha mégis szeretnénk ezt a lehetőséget is megfelelően lekezelni, minden állapotból a megfelelő tranzíción keresztül egy hibakezelő állapotba kell elvezetni a liftet, majd onnan vissza.



A lift egyetlen emelet szemszögéből megfigyelve

A mozgást a nyugalmi állapotból kiindulva figyeljük meg *(ezt szimbolizálja az ott elhelyezett token)*. Az, hogy az egész hálóban egyetlen token kering, azt mutatja, hogy a lift egy időben mindig egy és csakis egy állapotban lehet.
Ha valahová elhívják a liftet *(vagy beszáll egy utas, és ő utasítja mozgásra)*, az ajtó bezáródik **`Ajtot_zar`**, és a lift teljesíti a feladatot **`Mozog`**. Hogy pontosan hova megy, és útközben milyen egyéb feladatokat kap, az számunkra lényegtelen. Ha feladatát teljesítette, feltételezésünk szerint előbb-utóbb visszaérkezik erre a szintre **`Szintre_er`**, majd kinyitja az ajtót **`Ajtot_nyit`**. Ha a végrehajtandó utasítások tára üres, ezen a szinten marad **`Marad`**, és visszatér nyugalmi állapotba. Ha azonban még van mit tennie **`Tovabbhivas`**, újra bezárul az ajtó, és az előzőekhez hasonlóan folytatódik a lift tevékenysége.

Lássuk most, mi történik, ha a liftet teljes valójában szeretnénk szemlélni, azaz nem csupán egyetlen emelet szemszögéből, hanem teljes viselkedésére kíváncsiak vagyunk. Vegyünk egy kétemeletes épületet. Mivel ennek három szintje van *(földszint, 1. emelet, 2. emelet)*, három, az előzőhöz hasonló alhálóból épül majd fel a modellünk. A három részháló egymás mellé helyezése után egyedül azt kell megoldanunk, hogy ezek a megkövetelt funkcionalitásnak megfelelően össze is legyenek kapcsolva.
Először is azt kell megoldanunk, hogy ha egy adott szinten bezáródik az ajtó, a következő állapotban valamely szomszédos emeletre érkezzen meg a lift. Erre szolgálnak a következő tranzíciók: **`Felmegy_0-1, Lemegy_1-0; Felmegy_1-2, Lemegy_2-1.`** A tranzíció neve természetes módon utal a mozgás irányára, a benne szereplő számok pedig értelemszerűen azt jelzik, mely két emelet között történik a mozgás.
Azt az esetet azonban még így sem kezeltük le, amikor valamely szinten egyszerűen csak áthalad a lift megállás nélkül. *(Képzeljük el, milyen kényelmetlen lenne, ha a valóságbeli liftek úgy működnének, hogy minden egyes emeleten megállnának, kinyitnák ajtajukat, majd függetlenül attól, hogy van-e ki-és/vagy beszálló utas, mindig várnának egy bizonyos időt, s csak utána zárnák ajtajukat és indulnának tovább.)* Erre az esetre újabb két tranzíciót kell bevezetnünk, a **`Halad_le_1-0`**, és a **`Halad_fel_1-2`**-t. Ellentétes párjukra azért nincs szükség, mert például a földszint és az emelet között egészen biztosan csak földszinti ajtózáródás után tud haladni, hiszen a földszint alatt nincs több szintünk.
A következő oldalon látható árba ezt a kibővített, háromszintes épületre vonatkozó lift Petri-hálós modelljét mutatja be. Joggal merül fel a kérdés, hogy hogyan bővíthető tovább ez a modell három-, négy- esetleg több emeletes épület liftjének kezelésére. Bővíthető-e egyáltalán?
A válasz természetesen igen. Mégpedig oly módon, ahogyan az egyetlen szintből kiindulva konstruáltuk a háromszintes modellt: a második emeletről felfelé történő mozgást például első lépésként modellezzük egyetlen **`Mozog_2`** nevű tranzícióval, amelynek bemenő helye az **`Ajtot_zar_2`**, kimenő helye pedig a **`2_emeletre_er`** hely. Következő lépésként ezt a tranzíciót helyettesítjük a megfelelő rész-hálókkal, az igényelt szintek számának megfelelően.



A liftautomata Petri-hálója

De vajon „jó"-e a fent megalkotott Petri-háló? És egyáltalán: mit értünk a háló jóságán?

Egyik fontos kritérium lehet, hogy a háló holtpont-mentes legyen, azaz ne tartalmazzon olyan tranzíciót, amely bizonyos feltételek mellett „befagyhat", azaz nem tud üzemelni. Ha megnézzük a liftautomata Petri-hálóját, azt láthatjuk, hogy minden tranzíciónak egyetlen bemenő helye van, így biztosak lehetünk abban, hogy amennyiben a token ezen az ős-helyen van, a tranzíció mindenképp engedélyezett lesz, nem „ragad be" a működés.

Másik lényeges szempont, hogy adott állapotból *(tokeneloszlásból)* kiindulva elérhető-e minden hely, vagy vannak elszigetelt régiói a hálónak, amelyek semmilyen token-áramlással sem hozzáférhetőek.

A mi Petri-hálónk összefüggő, így elszigetelt régiói nincsenek. Most már csak arról kell meggyőződnünk, hogy bármely helyén van is aktuálisan a token, onnan bármely másik helyre újra eljuthat. A háló e tulajdonságát élőségnek nevezzük.

Amennyiben az egytokenes Petri-hálót egy irányított gráfnak fogunk fel, a probléma átfogalmazódik arra a kérdésre, hogy hány erősen összefüggő komponensből áll: ha egyből, akkor a háló élő, ha többből, akkor nem az.

Egy irányított gráfról akkor mondjuk, hogy erősen összefüggő, ha bármely pontpárjára létezik olyan irányított út, amely az egyik pontból indul, és a másikban végződik *(és fordítva)*. Olyan ez, mint ha egy képzeletbeli városban csupa egyirányú útjaink lennének, és autóval próbálnánk eljutni az A helyről B-be. Nyilván ez nehezebb probléma, mint gyalogosan, akikre nem vonatkoznak az egyirányúságra vonatkozó szabályok.

Lássunk most egy módszert az erősen összefüggő komponensek meghatározására! Mélységi bejárással járjuk be az irányított gráfot, s közben minden pontjához rendeljünk egy sorszámot. Ez a sorszám legyen az úgynevezett befejezési szám, amely azt mutatja, hogy az adott pontot hányadikként fejtettük ki teljesen *(Egy pont akkor van teljesen kifejtve, ha már az összes gyerekét bejártuk)*.

Készítsük el a gráf fordított gráfját, azaz fordítsuk meg minden él irányítását. Végül ezt a fordított gráfot járjuk be mélységi bejárással. Ha ez utóbbi bejárás során új gyökérpont választása nélkül be tudjuk járni a gráf összes pontját, a gráf egyetlen erős komponensből áll, azaz a Petri-hálónk *(ahonnan kiindultunk)*, élő.

A Petri-hálók még számos további tulajdonsággal jellemezhetők és analizálhatók, ezúttal ezekre nem térnék ki. Annyit jegyeznék meg csupán, hogy a ma forgalomban lévő modellező eszközök széleskörűen támogatják a különféle analíziseket, így gyakorlatilag semmit nem kell manuálisan elvégeznünk, a háló konstrukciója után pillanatok alatt megkaphatjuk annak vizsgálatakor született eredményeket.

Sok sikert, szerencsés próbálkozásokat kívánok mindenkinek!

Molnár Ágnes
agi@harpia.homeip.net

Internet Exlporer javítás

Az Internet Exploreren két újabb biztonsági rést javíthatunk a 2003. február elején megjelent MS03-004-es számú hotfixszel. Mint már megszokhattuk, ez a javítás tartalmazza az Explorerhez eddig kiadott javításokat, így ha valaki eddig elmulasztotta volna betömködni a már ismert réseket, az egyszerre megteheti az [msdownload] helyről letölthető biztonsági javítással.

A hibákat az Internet Explorer cross-domain biztonsági modelljében találták, melynek a feladata az lenne, hogy a különböző domainekben megnyitott weboldalak ne oszthassanak meg információkat. Azonban a hiányos biztonsági ellenőrzés következtében az Internet Explorer lehetővé teheti, hogy egy párbeszédablak használatával különböző domainben nyitott weboldalak ugyanazokhoz az adatokhoz férjenek hozzá.

Egy rosszindulatú támadó csak készít egy weboldalt, mely éppen erre a résre épít, majd a gyanútlan felhasználót megpróbálja rávenni, hogy az adott oldalt meglátogassa. Amint a felhasználó megnézi az oldalt, a támadó egy párbeszédablak segítségével futtathat egy scriptet, melynek segítségével információkhoz férhet hozzá. Megtudhatja melyek azok a rendszerfájlok, melyeket a felhasználó vagy az operációs rendszer nem használ, mi a fájlok neve és mi a teljes elérési útjuk. Hozzáférhet bármilyen adathoz, melyet más oldalakkal megosztottunk. A legrosszabb esetben akár azt is elérheti, hogy az adott scriptet ne csak futtassa, hanem el is helyezze azt a felhasználó rendszerében. Ezen felül e résen keresztül a támadó futtatható állományokhoz is hozzáférhet.

Egy ezzel rokon rés lehetőséget ad arra, hogy az Internet Explorer showHelp() függvényét a megfelelő biztonsági azonosítás nélkül futassuk. A showHelp() egyike azoknak a súgómetódusoknak, mellyel HTML-oldalon jeleníthetjük meg egy súgófájl tartalmát. Ez a függvény több protokollhoz fér hozzá, mint amennyi szükséges, és ez potenciális lehetőséget nyújt egy támadónak, hogy felhasználói információkhoz férjen hozzá, futtatható állományokat hívjon meg, vagy rosszindulatú kódot töltsön egy felhasználó rendszerébe.

Ebben az esetben is rá kell venni valakit, hogy látogasson meg egy adott oldalt az Interneten. Aztán a látogató gépén egy ismert fájlt megnyit a támadó egy showHelp ablakban. Erről a fájlról az összes elérhető információt elküldi egy második showHelp ablaknak egy ravasz URL-en keresztül, így szerezve meg adatokat.

A biztonsági hiba folytán a támadó hozzájuthat egy oldal által tárolt cookie-hoz, a felhasználó gépén található fájlok tartalmát is megismerheti, sőt a HTML Helpek által támogatott shortcut függvénnyel akár paraméterezett utasításokat is futtathat. Így ha böngészés közben hirtelen elindul az aknakereső, valószínűleg támadás áldozatai lettünk, és itt az ideje a biztonsági rendszerünkön tátongó lyukakat befoltozni.

Az alábbi példa mutatja, milyen könnyen hozzáférhetünk akár egy cookie tartalmához is:

```
<script>
showHelp("file:");
showHelp("http://www.netacademia.net");
showHelp("javascript:alert(document.cookie)");
</script>
```

Eredményül egy sütit, és egy Help-ablakba szorított weblapot kapunk:

ASP munkamenet sütije

Ha ezt a javítást telepítjük, a window.showHelp() függvény elérhetetlenné válik. Telepítsük a legutolsó HTML Help Update-t, és a showHelp() újra meghívható, működik. Hozzá kell azonban tenni, hogy ekkor már bizonyos korlátozások lépnek életbe a függvény használatakor.

- A showHelp csak a támogatott protokollokat használhatja, amikor egy weboldalt vagy egy súgófájlt *(.chm)* nyit meg.
- A HTML Help támogatja a shortcut függvényt, mellyel különböző utasításokat készíthetünk Például, ha egy HTML Helpben egy adott szóra kattintunk, akkor elindítja a NotePadot, vagy éppen nyit egy új információs ablakot. Ez a lehetőség a javítás telepítése után már nem működik, ha a súgót a showHelp() függvénnyel hívtuk meg. Ha a felhasználó dupla kattintással nyitja meg ugyanezt a fájlt, a shortcut újra használható. Akkor sincs megkötés, ha egy alkalmazás súgóját használjuk.

Végül nézzük, az Internet Explorer mely verzióihoz adták közre ezt a javítást:

Internet Explorer verzió	Platform
IE 5.01 SP3	Windows 2000 SP3
IE 5.5 SP2	Windows Milennium
IE 6.0 Gold	Windows XP Gold
IE 6.1 SP1	Windows XP SP1, Windows 2000 SP3, Windows NT 4.0 sp6a, Windows Millenium, Windows 98

A telepítés után a gépet újra kell indítani. A javítást ezek után eltávolítani nem lehet!

Borsi Katalin
bobo@netacademia.net

A bitlegelők tragédiája

Egyszer volt, hol nem volt, volt egyszer sok tehénpásztor. Az egyik arra gondolt, hogy több tehén többet tejel, ezért megnövelte a csordája létszámát. Kiderült, hogy igaza lett, mire a szomszéd pásztor is hasonlóképp tett. Egy bolond százat csinál, de még a felénél sem tartottak, amikor annyi tehén bőgött a réteken, hogy már egyikük sem lakott jól teljesen...

Emiatt persze kevesebb tejet adtak, így akik még tovább bővítették a csordájukat, csak a többiek rovására tudtak javítani az eredményükön. Végül annyi jószág legelészett, hogy már az is rosszul járt, aki magának vásárolt tehenet. Ez az együttműködésre képtelen, oktalan pásztorok egyensúlyi helyzete.

E problémát elsőként Garrett Hardin biológus vázolta fel 1968-ban **[population]**. Az elmélet oly frappánsan írja le az embereknek a közjavakkal szemben tanúsított viselkedését, hogy más tudományágak is sikerrel alkalmazták, és alkalmazzák a mai napig. Az Internet például nagyon hasonlít egy közlegelőre, amelyen kódpásztorok milliói legeltetik programjaikat. A sávszélesség gátlástalan bitorlásának következményével, a torlódással, ilyen vagy olyan formában már bizonyára mindenki találkozott. A torlódásvédelem egyidős az Internettel, és a kapacitásnál gyorsabb ütemben növekvő felhasználás miatt *(amit gúnyosan nevezhetnénk akár multimédiás őrületnek is)* egyre inkább a figyelem középpontjába kerül.

Az Internet protokolljai pillanatnyilag nem dúskálnak a torlódásvédelmi lehetőségekben. A TCP időtúllépésével, illetve csomagvesztésével gyakorlatilag ki is merítettük a sort. Dehát akkor mit lehet tenni? Ezegyszer végre – a megszokott iránytól eltérően – a közgazdaságtudomány siethet a műszaki élet segítségére. A kereslet-kínálat kézenfekvő következménye, hogy az erőforrások felhasználása szabályozható az ár változtatásával. Ez a trükk köszön vissza például a telefondíjak időzónásításánál, az olcsóbb éjszakai áramnál, vagy egyes országok autópályadíjainál. Ilyenkor a terhelés kiegyenlítése a cél, mivel műszaki-gazdasági szempontból általában ez a legkedvezőbb. Az Internet vagy egy Intranet sem különb. Nem keveredtünk itt ellentmondásba? Ha szigorúak akarunk lenni, a válasz igen. A problémafelvetés ugyanis a közjavakkal kapcsolatban történt, ha viszont valaminek árat szabunk, akkor az elveszti "köz" jellegét. Csakhogy a költséget az internetező nem biztos, hogy közvetlenül díj fizetésével viselné, elképzelhető mindez a rendelkezésére bocsátott sávszélesség nagyságának korlátozásával. A kevésbé kötekedő felelet tehát lehet nem is.

Lássuk egy egyszerű példán, mit jelent mindez? Egy vállalatnál előnyben részesíthetjük a videokonferenciákat és az igazgató webezését a többi alkalmazással és felhasználóval szemben, ami abban nyilvánul meg, hogy ezek a beállításoktól függő mértékben nagyobb sávszélességet kapnak. Ha azonban a kiemelt alkalmazások egyike sem fut, a felszabadult sávszélességet a többi alkalmazás minden további nélkül használhatja. Ugyanez az elv egyetlen számítógép és felhasználó esetében is alkalmazható, például úgy, hogy az elektronikus levelek küldésének nagyobb prioritást adunk, mint a webezésnek vagy az FTP letöltéseknek.

A Resource Pricing modell bevezetése az Interneten már a 90-es évek eleje óta napirenden van, és a Microsoft Research hálózati csoportja is próbálkozik ennek megvalósításával **[cows]**. Peter Key kidolgozott egy olyan modellt, amelyben a túlterhelődő erőforrások *(processzor, memória, tápegység, sávszélesség stb.)* ára emelkedik, és így elvben akár az egész Internetre alkalmazható lenne. Az alapötlet szerint torlódás kialakulásánál az útválasztók megjelölik a csomagokat. Ezek a jelek lehetnek nagyon egyszerűek, mint például az IETF Explicit Congestion Notification *(ECN)* ajánlásában *(RFC 3168)*, ahol mindössze egyetlen bit jelzi a forgalmi dugó kialakulását, de lehetnek ennél sokkal kifinomultabbak és értelemszerűen bonyolultabbak is. Okosan megválasztott költségfüggvénnyel befolyásolható a sávszélesség felhasználása. Ha a költség pénz formájában nyilvánul meg, magától értetődő, hogy a felhasználókat mi ösztönzi a játékszabályok betartására. Ha mégsem, akkor elképzelhetők például olyan alkalmazások, amelyek korlátozott mennyiségű jelet használhatnak fel adott idő alatt, és próbálnak abból jobban gazdálkodni, mint mi a havi fizetésünkből. A torlódásjelzés információtartalmának nagysága vitatott téma. Az ECN megkérdőjelezhetetlen előnye, hogy az IPv4 fejléc Type of Service mezőjének két kihasználatlan bitjét használja fel. Ez azonban meglehetősen kevés például a torlódás mértékének vagy helyének a jelzésére. Több bites leírásra alkalmas lesz az IPv6, de ha addig nem bírjuk cérnával, az operációs rendszerekbe építhető torlódáskezelő is megoldás lehet. Nemsokára valószínűleg a Windowsokban is találkozhatunk vele.

A modell bevezetése az Interneten azonban egy komoly társadalmi kérdést vet fel: mi alapján állítunk fel sorrendet az emberek között? Richard Black sem tett kísérletet a válaszra, helyette a vállalati és otthoni hálózatokra készített prototípussal tesztelte Key modelljének használhatóságát, pozitív eredménnyel. A sikereken felbuzdulva Key és Dinan Gunawardena néhány népszerű hálózati alkalmazáson, többek között a Windows Median és a Windows Messengeren is kipróbálta az ötletet. Működött. A kép- és hangminőség például a hálózat terheltségének függvényében változott, biztosítva ezzel az adás folyamatosságát.

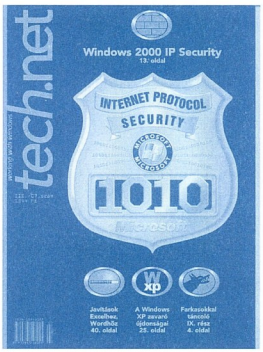
Hardin mély meggyőződéssel vallja, hogy a közlegelő típusú problémáknak nincs technikai értelemben vett megoldása, ez csak és kizárólag az erkölcsi normák megváltoztatásával lehetséges **[population]**.

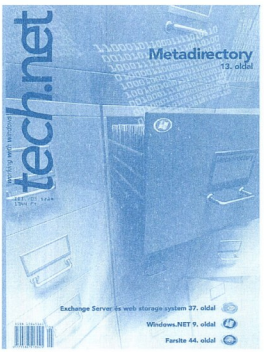
Zacco@fw.hu

tech.net magazin előfizetési szelvény

fax

Címzett: NetAcademia Kft.
Faxszám: (1) 472-1215





http://technet.netacademia.net/subs/szamrend.asp • e-mail: terjesztes@netacademia.net • fax: (1) 472-1215

Előfizetem a **tech.net** magazint:

..... példányban egy évre (14.784 Ft)
..... példányban fél évre (7.392 Ft).

Megrendelő neve: .....

Cég neve: .....

Számlázási cím: ☐☐☐☐ .....

.....

Postázási cím: ☐☐☐☐ .....

.....

E-mail cím: .....

Telefon: .....

Fax: .....

A fizetés módja: ☐ csekkel ☐ átutalással

Kelt: ...../ ...../ .....

Aláírás: .....

MesterQrzus megrendelőlap

fax

Címzett: NetAcademia Kft.
Faxszám: (1) 472-1215

A 2003 első félévében megrendezésre kerülő MesterQrzus konferenciák:

2003. március 27. 13:45 - 18:00 **5000 Ft+ÁFA/fő**

Fóti Marcell: **Bevezetés a hálózati forgalom elemzésébe**
NetLock Kft. vendégelőadó: **A nyílt kulcsú technológia építőkockái**
Fülöp Miklós: **Nyílt kulcsú architektúra a Windowsban**
Gyakorlat: Felhasználói és kiszolgálóoldali PKI-gyakorlatok
Ajándék: Hiteles NetLock tanúsítványpár elektronikus aláíráshoz és titkosításhoz. A kulcstároló eszközök a helyszínen megvásárolhatók.

2003. május 29. 13:45 - 18:00 **5000 Ft+ÁFA/fő**

Fóti Marcell: **Az elektronikus levelezés védelmének lehetőségei Exchange Serveren**
Soczó Zsolt: **SOAP**
VirusBuster Kft. vendégelőadó: **A Windows rendszerek támadható, és védelmi felületei**
Gyakorlat: Felhasználói és kiszolgálóoldali PKI-gyakorlatok
Ajándék: Virus Buster Personal Edition, 1 éves előfizetéssel

A konferenciákkal kapcsolatban a változtatás jogát fenntartjuk!

Megrendelem a MesterQrzus konferenciákon való részvételt az alábbiak szerint:

2003. március 27. ..... **fő**
2003. május 29. ..... **fő**

Megrendelő neve (kapcsolattartó): .....

Cég neve: .....

Számlázási cím: ☐☐☐☐ .....

.....

E-mail cím: .....

Telefon: .....

Fax: .....

Kelt:...../...../..... Aláírás: .....

Bővebb információt a **http://www.netacademia.net/mesterq** honlapon talál, illetve a következő elérhetőségeinken kérhet:
Tel.: 472-1214 Fax: 472-1215 info@netacademia.net

A hálózatok csapdájában...

Ha úgy érzi, a hálózatok csapdájába esett és a helyzet reménytelen, jöjjön el a SZÁMALK Továbbképzés Cisco tanfolyamaira, ahol megtalálhatja a megoldást...

A **SZÁMALK Továbbképzés** saját fejlesztésű és tematikájú, hivatalos Cisco anyagokra épülő tanfolyamokat indít, elsősorban a hálózati valamint hang- és adatátviteli technológiák területen.

A magas szintű hálózati ismeretek nélkülözhetetlenek az összetett informatikai rendszerek üzemeltetéséhez és hivatalos **Microsoft képzéseink** hatékony elsajátításához is.

Kedvező ár, korszerű hardver- és szoftverkörnyezet, egyedileg összeállított hallgatói jegyzet, eredeti segédanyagok, minősítő vizsgával rendelkező oktató.

Tavaszra meghirdetett tanfolyamaink:

Cisco kapcsolt hálózatok:	**április 7–11.**
Cisco hálózati alapismeretek és eszközök:	**április 14–18.**
Cisco skálázható hálózatok kiépítése, tervezése:	**május 19–23.**

A tanfolyamokkal kapcsolatos további információkért, teljes kínálatért kérjük keresse munkatársainkat, látogassa meg internetoldalunkat! Figyelje tanfolyamokhoz kapcsolódó akcióinkat is!

Legyen Ön is hivatalos szakértő! Tanfolyamaink a hivatalos vizsgákra való felkészülésben is segítenek. Nemzetközi minősítést szerezhet, akár egy vizsgával is!

Egészítse ki tanfolyamainkon szerzett ismereteit hivatalos szakkönyvekkel!

A SZÁMALK Továbbképzés a Microsoft és Cisco által forgalmazott több száz kiadványt, elektronikus alapú tananyagot és szakkönyvet kínál az általános hálózati technológiákkal foglalkozó kiadványoktól a specifikus, üzemeltetési kézkönyveken át, a hivatalos vizsgákra és minősítésekre felkészítő tréning csomagokig és vizsgafelkészítő tananyagokig.
Folyamatosan bővülő termékskála, kedvező árak, nagy raktárkészlet.

Cisco Systems

Microsoft CERTIFIED Technical Education Center

SZÁMALK TOVÁBBKÉPZÉS

SZÁMALK

☎ 203-0304/3050 (Simon Ferenc) **www.szamalk.hu/tisza** 1115 Budapest, Etele út 68.