tech.net

working with windows

800 számítógép átvizsgálása fél nap alatt

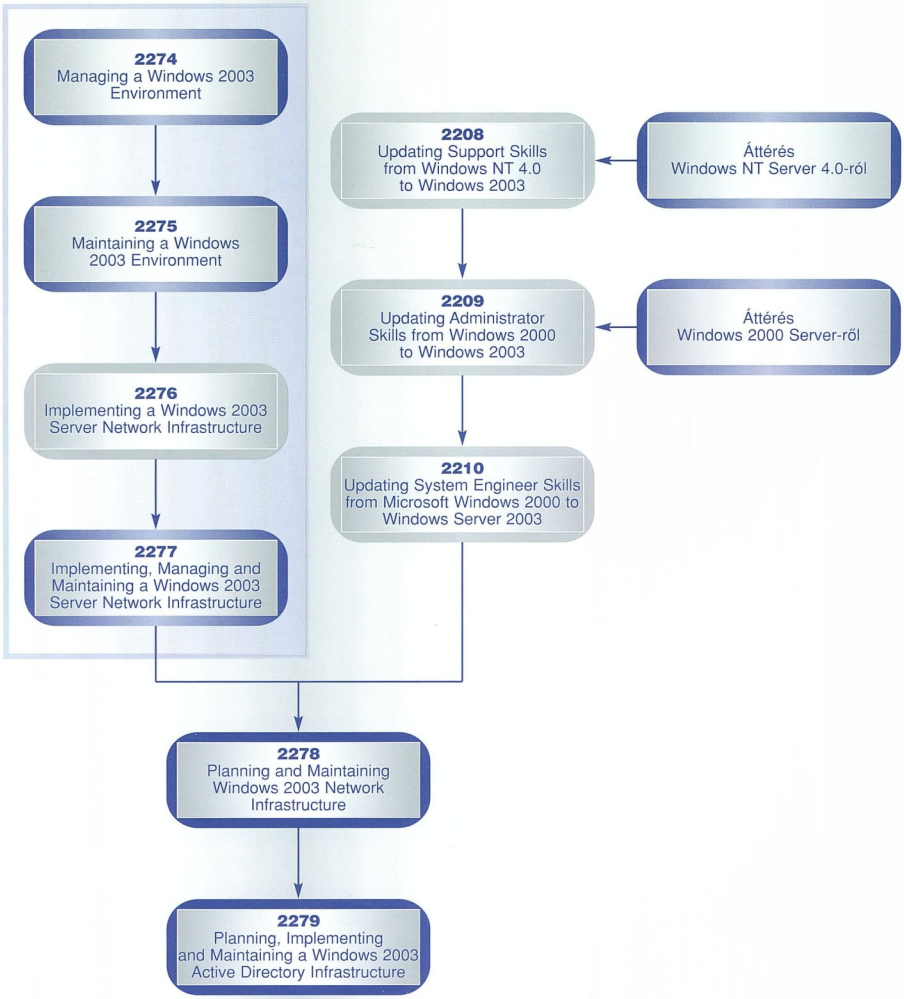
IV. / 10. szám
2003. október
1344 Ft

ISSN 15865185 10
9 771586 518005

netACADEMIA

A LEGJOBBAKAT TANÍTJUK

Windows 2003 Server rendszergazdai tanfolyamok térképe



www.netacademia.net • 1062 Budapest, Andrássy út 62. • Telefon: (1) 472-1214 • Fax: (1) 472-1215

Szerkesztőség:
Főszerkesztő: **Fóti Marcell**
marcellf@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.
1033 Budapest,
Csikós utca 8.
Felelős vezető: **Tóth Béláné**

ISSN 1586-5185

Amiről nem tudunk, az nincs?

Mint tudjuk, az Active Directory DNS alapokon nyugszik. Mégpedig a forward lookup zóna megfelelő bejegyzésein. Aki nem adja meg az AD-nak, ami DNS-ügyben jár neki, az elveszít mindent, a címtár nemhogy kívülről nem érhető el, de önmagát sem találja meg. Ez idáig triviális.

De mire való a reverse lookup zóna? Tehát az a zónatípus, amelyik nem név->IP, hanem pontosan fordítva, IP->név feloldást végez. *(Ennek a „névfeloldásnak" az alapja az in-addr.arpa zóna alá felvett, az IP-címből képzett zónában található PTR record.)* Ha jól megvizsgáljuk a Windows működését, azt tapasztaljuk, hogy erre semi szükség. Nincs olyan Windows-komponens, amelyik revesre lookup lekérdezést végezne. Ebből a tapasztalatból kiindulva magam is hangoztattam, hogy reverse lookup zónára nincs szükség, tehát ne is fárasszuk magunkat annak létrehozásával.

De nemrégiben rájöttem, hogy ez a gyakorlat milyen súlyos biztonsági problémához vezet. Tűzfal ide-tűzfal oda az összes Windows-munkaállomás neve és belső IP-címe publikálásra kerül az Internet felé! Ebből az adathalmazból pedig fantasztikusan pontos hálózattérképeket készíthetnek arra illetéktelen személyek. Tudni fogják mind a munkaállomások, mind a kiszolgálók nevét, és belső IP-címét. Mindez azért van így, mert a Windows 2000 és utódai *(XP, YP, 2003, Longhorn, 3000)* IP-cím felvételkor és változáskor, valamint újraindításkor alapértelmezésben kibocsátanak magukból egy *(-két)* dinamkus DNS kérést, hogy a DNS Server regisztrálja be a zónafájlba az új IP-címüket. Valójában két DDNS-kérés indul el a hálón: egy forward és egy reverse. *(Mégegyszer emlékeztetőül: a forward névhez rendel IP-t, míg a reverse: IP-hez rendel nevet.)*

A forward DDNS általában célt ér, és befut a céges DNS Server megfelelő zónájába. A reverse DDNS azonban *(hisz nincs is ilyen zónánk)* a szokásos *(rekurzív)* módon kimegy a root name serverekhez, onnan az arpa-hoz, onnan az in-addr.arpa-hoz stb. És mit tartalmaz? Hogy a 172.16.12.12 cím a MARI_NENI_GEPE című hoszthoz tartozik. Idővel, szép fokozatosan minden PC neve és IP-címe kikerül a netre! Ez azt hiszem, eléggé kellemetlen mellékhatás ahhoz, hogy elgondolkodjunk a megfelelő reverse DNS zóna létrehozásán. És azt is javaslom, hogy aki korábban azt hangoztatta, hogy ilyen zóna nem kell, azt dobáljuk meg záptojással és paradicsommal. Tehát engem.

Kimértem Network Monitorral is, vajon pontosan hová igyekeznek a belső IP-címeket *(10, 172 és 192)* tartalmazó reverse lookup DDNS-utasítások. Börtönbe. Az elutasító választ a prisoner.iana.org cím adja.

Fóti Marcell
MCT
marcellf@netacademia.net

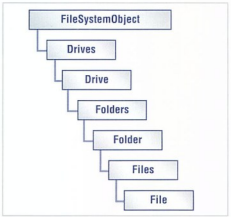
2003. 10. szám

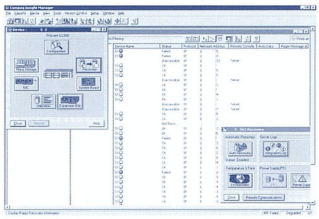
Scripting a gyakorlatban

800 számítógép átvizsgálása fél nap alatt

Egy hazai nagyvállalatánál jelenleg térnek át a munkaállomásokon NT4-ről Windows XP-re. A teljes újrainstallálás mellett döntenek, hogy az NT4-ben hat év alatt felgyülemlett szemét ne koszolja össze a friss operációs rendszert. Biztosra megyünk: a C:\ formázásával garantáltan eltűnik az NT4 – de a felhasználók adatai is. Azokat tehát előbb le kell menteni. De hol tárolják a dokumentumaikat? Ezt kellene gyorsan és biztosan kideríteni!

4. oldal





Rendszerfelügyelet nagyvállalati környezetben

Avagy a dromedár szemfüle

Ahogyan az informatika egyre nagyobb teret követel magának a vállalati környezetben, úgy egyre komplexebbé válnak a rendszerek, egyre nagyobb rendelkezésre állást követelnek meg tőle, így egyre nagyobb az igény a rendszer állapotának nyomonkövetésére. A rendszerfelügyelet itt nyújt hathatós segítséget az üzemeltetés részére.

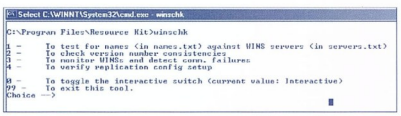
8. oldal

Gyógyszeresdoboz

Windows 2000 Server Resource Kit V.

Folytassuk tovább kisebb-nagyobb hálózati segédprogramokkal.

11. oldal



A Windows 2003 TCP/IP-implementációja

A TCP/IP-verem felépítése és főbb funkciói

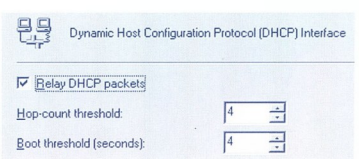
A TCP/IP az Internet működésének alapja, és mára a vállalati hálózatokban is a legnépszerűbb hálózati protokollá vált. Útválasztási funkciói maximális rugalmasságot biztosítanak a nagyvállalati hálózatokban is. A TCP/IP-verem megvalósítása és szolgáltatásai tehát döntő fontosságúak minden operációs rendszer hálózati funkcióinak felépítésében.

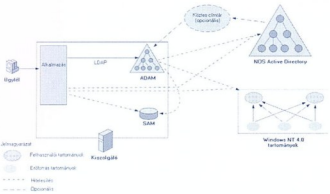
13. oldal

A DHCP rejtett szépségei – V.

A címkiosztó szolgáltatás üzembehelyezésekor különösen fontos jól megválasztani a helyszínt. Egyszerű környezetben, ha nincs is több telephely, ez nem probléma, ma azonban már egyre több összetett hálózat működik. Lehet a topológia bármilyen bonyolult, a DHCP szervereknek megfelelően kell működniük.

17. oldal





Active Directory Application Mode

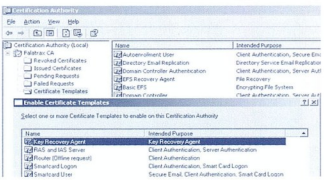
Címtárat mindenkinek!

Az ADAM használatával többé nincs szükség egyedi címtárszolgáltatásra speciális, esetleg belső fejlesztésű alkalmazásaink számára sem, mivel az ADAM messzemenően képes alkalmazkodni bármilyen feladathoz. Minden ezt igénylő alkalmazás önálló, de az Active Directory-val együttműködő címtárat használhat egyedi sémával és replikációs beállításokkal.

20. oldal

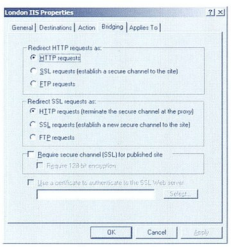
Tanúsítványkiadók a Windowsban

Kulcsarchiválás és helyreállítás, adatmentés, Certificate Trust List



A Windows Server 2003, Enterprise Edition tanúsítványkiadó szolgáltatása segítségével biztonságosan tárolhatók és szükség esetén vissza is tölthetők a felhasználók privát kulcsai. Cikkünkben kitérünk még a tanúsítványkiadó adatbázisának biztonsági mentésére, valamint a tanúsítványláncok összekapcsolásának egyik módjára.

24. oldal



Háromlábú ISA Server

DMZ külön hálózaton

Amikor egy cég „magához ragadja" az Internetes szolgáltatások körét, és saját SMTP, HTTP és egyéb kiszolgálókat üzemeltet, mindig felmerül az a nem is egyszerű tervezési kérdés, hogy hova tegyük a kiszolgálókat? A belső hálózatra, és onnan publikáljuk? Ez a jelenlegi, férges korszakban veszélyes lehet. Két tűzfal közötti DMZ-be? De hisz az plusz egy vasat igényel! Vagy netán használjuk a háromlábú modellt? Miért ne?

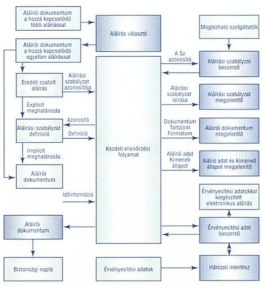
28. oldal

Biztonságos aláírás kezelő alkalmazás készítése V.

Az aláírás-ellenőrző alkalmazás

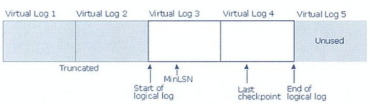
Az előző részben az aláírás kezelő alkalmazás adatai, valamint az aláírás létrehozó alkalmazás ezeket hasznosító komponensei, s az ezekkel szemben támasztott követelmények lettek ismertetve. Most az aláírás-ellenőrző alkalmazást mutatjuk be hasonlóképpen. Ezzel nagy ívű sorozatunk egyben a végére is ér. Bár nem fért bele részletesen minden, a lényeget sikerült belegyömöszölni. Aki a leírtakat magáévá tudta tenni bátran nekivághat egy aláírás kezelő alkalmazás fejlesztésének vagy bevezetésének.

31. oldal



A tranzakciónapló és környéke

SQL Server 2000 adat- és tranzakciónapló-fájlok, adatbázis helyreállítási modellek, mentési stratégiák



Az adatok rendszeres mentésének fontosságát mindenki tudja. Arról már olykor elfeledkezünk, hogy felmérjük, mennyi tranzakció (új adat, vagy módosítás) elvesztését tudjuk elviselni. Az SQL Server tranzakciónaplóját sok rendszergazda „nem szereti". Gyakori kérdés, miért nem lehet a tranzakciónaplót kikapcsolni? Meglepően sok adatbázis „simple" helyreállítási modellben működik, ami a tranzakciónapló automatikus ürítésével jár. Ez (többnyire) megakadályozza, hogy a logfájl nagyra nőjön, de ugyanakkor lemezhiba, vagy egy véletlen táblatörlés után lehetetlenné teszi az utolsó pillanatig érvényes helyreállítást.

36. oldal

Scripting a gyakorlatban

800 számítógép átvizsgálása fél nap alatt

Egy hazai nagyvállalatánál jelenleg térnek át a munkaállomásokon NT4-ről Windows XP-re. A teljes újrainstallálás mellett döntenek, hogy az NT4-ben hat év alatt felgyülemlett szemét ne koszolja össze a friss operációs rendszert. Biztosra megyünk: a C:\ formázásával garantáltan eltűnik az NT4 – de a felhasználók adatai is. Azokat tehát előbb le kell menteni. De hol tárolják a dokumentumaikat? Ezt kellene gyorsan és biztosan kideríteni!

A feladat tehát egy olyan lista készítése, amely minden létező helyről tartalmazza a felhasználó adatait, még akkor is, ha az elmúlt hat évben a SYSTEM32 könyvtárba mentett. Szélsőséges példa az a felhasználó is, aki minden könyvtárának úgy adott nevet, hogy belecsapott a billentyűzetbe. Így lett adatainak főkönyvtára az „asdfasd", alatta pedig rendre „dfgsfe" és „qweqwe" könyvtárakat találunk.

A feladat elvégzéséhez valamilyen leltárra volna szükségünk, de egy olyan okos leltárra, ami az ál-dokumentumokat *(eula.doc és társai)* nem tartalmazza. Szintén nem kell a dokumentum, ha a kukában, vagy más lehetetlen helyen tanyázik. S ha már megvannak, következő lépésben másoljuk is át őket a D:\ meghajtóra, amit nem fogunk megformázni. Ott biztonságban lesznek a frissítés alatt is.

Ez utóbbi, tehát a mentési igény kapásból kiüti a kezünkből a hagyományos rendszerfelügyeleti eszközöket, mert leltárt készíteni még tudnak, még arra is képesek, hogy a hamis dokumentumokat kiszűrjék, de ezek lokális lementésére önmagukban már nem képesek. Ahhoz már valami kiegészítő alkalmazás szükséges.

Hagyományos eszközökkel kísérletezünk

Első megközelítésben kipróbálhatjuk, mit tud maga a jó öreg DIR parancs. Megadja a feleletet, hol bújnak meg a dokumentumok? Próbáljuk ki ezt:

```
DIR *.DOC /S
```

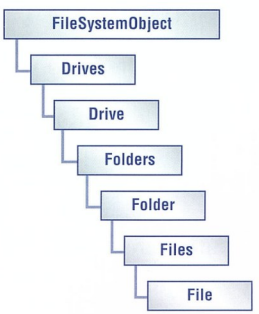
Igen, igen, de az eredmény nem kielégítő, hanem inkább szörnyű. Jártunk a kukában, kilistáztuk az összes winword.doc-ot *(most derül csak ki, hogy hány tucatszor van fenn a gépen)*, kilistáztuk a Windows mélyén megbúvó dokumentumokat, egyszóval kaptunk egy vödör kacatot. Jó lenne, ha a DIR-nek meg tudnánk adni egy kivétellistát, hogy mit NE vegyen figyelembe. Ilyet sajnos az öreg jószág nem tud.

A fejlesztés első lépése: v0.82 – megszólal az FSO

Akinek kalapácsa van, mindent szögnek néz – mondja a régi kínai bölcselet. Aki tud scriptet írni, minden problémát azzal old meg. Ha a gyári DIR nem tudja lekezelni a kivételeket, majd a saját kézzel fejlesztett DURR.VBS megoldja a problémát. Dirdurr.

Döntés született tehát a DURR.VBS kifejlesztéséről, ami a DIR belső parancs utódja lesz. Ha készen van, könnyedén bele lehet fejleszteni a mentési képességet is.

Hogyan kezdjünk hozzá? A Windows Scripting Host rendszerben a FileSystemObject alkalmas a lemezen lévő adatok manipulálására. Tekintsük át röviden, milyen objektumokból áll ez az objektumtár? A következő ábra szemléletesen összefoglalja, milyen hierarchiában érhetők el a lemezen tárolt adatok. Ennek megfelelő objektumhierarchiát tartalmaz a FileSystemObject, emiatt nagyon kényelmes lesz majd a használata. Vágjunk is bele, írjuk meg a DURR.VBS-t, mely kilistázza a dokumentumokat a merevlemezről! Ehhez a funkcióhoz több lépésben jutunk el, először elégedjünk meg azzal, hogy egyáltalán szóra bírjuk a rendszert.



A FileSystemObject objektumcsalád

Az alábbi kód kilistázza a C:\ meghajtó gyökerében található mappák neveit.

```
1. option explicit
2. dim oFSO, oDrive, oFolder
3. set oFSO =
     createobject("scripting.filesystemobject")
4. set oDrive = oFSO.GetDrive("C:\")
5. for each oFOlder in
     oDrive.RootFOlder.Subfolders
6.   wscript.echo oFolder.Name
7. next
```

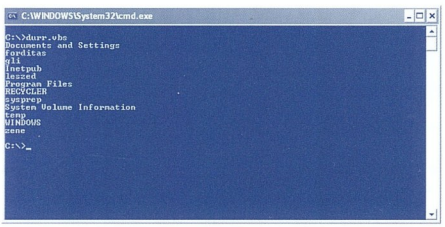
Magyarázat:

1. Az első sor szerepe, hogy egyszer s mindenkorra kizárjuk a deklarálatlan változók használatát. Ez feltétlenül szükséges, így ugyanis elejét vehetjük, hogy a „levegőből" képződjenek üres változók *(félregépelések folytán)*.
2. A második sorban deklaráljuk a felhasználni kívánt változókat. Adattípust nem adunk meg, mert olyan a

VBScriptben nincs. Minden változó Variant típusú. Ami a változóneveket illeti, roppant tudományosan néz ki az oDrive és az oFSO, de a valóságban ezek tetszőleges változónevek. Éppenséggel kutyulinak is hívhatnánk a mappaobjektumot. Abból aztán olyan kód keletkezne, amit két hét távlatából mi magunk sem lennénk képesek értelmezni. Ezért adtam a változóknak funkcionális nevet. A nevek elején az „o" hangocska pedig Simonyi Károly *(Microsoft)* találmánya, ezt hívják magyar jelölésnek *(hungarian notation)*. Az a szerepe, hogy egy pillantás alatt felismerjük a változó típusát *(o=objektum)*. Ez különösen hasznos a VBScript nyelvben, ahol nincsenek is valódi adattípusok.

3. Létrehozunk egy Scripting.FileSystemObject objektumot, melynek kezelőjét beletöltjük az oFSO változóba. A későbbiekben ennél a „nyélnél" fogva lehet őt rángatni.
4. Az oDrive változóba kerül a C:\ meghajtóobjektum. Ennek gyökérkönyvtárán keresztül vezet az út le a mappákig, fájlokig.

Az 5-6-7. lépésben ciklusban kiíratjuk a meghajtó gyökérkönyvtára alatti alkönyvtárak nevét. Az eredmény így néz ki *(cscript.exe futtatómotor használatával)*:



A DURR.VBS 0.82-es verziója kilistázta a gyökérkönyvtár alkönyvtárait

Ez megvolt, jöhet a következő lépés: a teljes könyvtárstruktúra bejárása.

DURR.VBS v0.85 – könyvtárfa bejárása

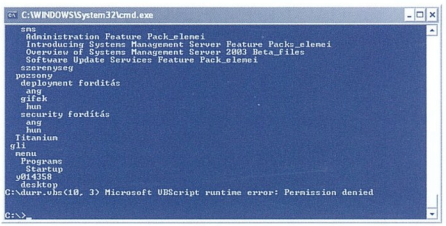
Tetszőleges mélységű faszerkezetek bejárására rekurzív függvényhívást használhatunk. A függvény elvégzi feladatát azon a könyvtáron, ahol éppen áll, majd meghívja önmagát a gyermekkönyvtárainak átadásával. A v0.82-es kódon tehát változtatnunk kell: a könyvtárnév-kiírási feladatot egy olyan függvénybe vagy eljárásba helyezzük át, ami nem mellesleg meghívja önmagát - saját gyermekeinek lekezelésére. Ez a kód így néz ki *(az első négy sor ugyanaz, ezt nem ismétlem meg)*:

```
 5. konyvtarlistazo oDrive.RootFolder, 0

 6. sub konyvtarlistazo(oFolder, szint)
 7. dim oSubFolder
 8.    wscript.echo space(szint) & oFolder.Name
 9.    for each oSubFolder in oFolder.Subfolders
10.      konyvtarlistazo oSubFolder, szint+1
11.    next
12. end sub
```

A főprogram ötödik sora ebben a változatban már csak meghív egy eljárást, és minden feladatot arra bíz. A szubrutin neve: konyvtarlistazo, és két bemeneti paramétere van. Elsőként azt a könyvtárobjektumot kell átadni neki, amelyikkel foglalkoznia kell, másodikként pedig egy szintszámlálót kap, amit a könyvtárfa kiírásánál használok fel a lista gyönyörűbbé tételére: minden alkönyvtár egy szóközzel beljebb kerül, mint a szülője.

A nyolcadik sorban kiírom a képernyőre az aktuális mappa nevét *(beljebbtolva a gyári space() függvényt által visszaadott darabszámú szóközzel)*. A 9-11. sorban a szubrutin rekurzív meghívását láthatjuk: minden alkönyvtárára ráeresztjük ugyanezt az eljárást. A rekurzió addig hatol lefelé, amíg talál alkönyvtárakat. Futtassuk! Ha átadhatnám, milyen vizuális élményt nyújt, ahogyan ez a nyúlfarknyi script bejárja az egész C:\ meghajtót! Csak gördül, gördül felfelé a sok-sok könyvtárnév! Gyönyörű! Naphosszat elnézném, de hopp, elszállt!



DURR.VBS v0.85. Jogosultsági hiba miatt elszállt

Úgy tűnik, hibakezelés nélkül nem lehet bejárni egy partíciót. Az új verzióban már ez is benne lesz!

DURR.VBS v0.93 – némi hibakezelés

A VBScript igen ostoba hibakezelési lehetőséggel rendelkezik, de céljainknak megfelel. Minden kritikus programsor elé be kell írni, hogy „ON ERROR RESUME NEXT", ennek hatására hiba esetén sem szakad meg a programocska futása. A hiba kódja bekerül az erre a célra létrehozott Err objektum Number tagváltozójába, amit a kritikus kódrészt követően azonnal illik megvizsgálni, vajon történt-e valami galiba?

A hibaüzenet tanúsága szerint a tizedik sor nem tudott lefutni, vagyis a mappa gyermekeit nem sikerült kilistázni. Ezt a részt vegyük körbe hibakezeléssel!

```
on error resume next
for each oSubFolder in oFolder.Subfolders
  if err.Number = 5 then
    wscript.echo "Access denied"
    err.Clear
  else
    konyvtarlistazo oSubFolder, szint+1
  end if
next
```

Fáradozásunkat siker koronázta, a script végigfut a teljes partíción. Igaz, hogy egyelőre csak a jogosultság hiányából fakadó hibákat kezeltük le, de egyelőre nem is találkoztunk más hibával. A világ összes hibájára ne próbáljunk azonnali megoldást találni.

Most már lassan jöhetne a tényleges munkavégzés, a hasznos dokumentumok megtalálása. A könyvtár nevének kiírása tájékán kellene meghívni egy másik eljárást, vagy függvényt ami az adott könyvtárból kiválogatja a hasznos fájlokat. Legyen a neve fajlvalogato!

DURR.VBS v0.97 – fájlok feldolgozása

Gondolkozzunk egy kicsit! Nekünk csak azokkal a mappákkal kell foglalkoznunk, amelyekben volt hasznos tartalom. E cél elérése érdekében a fajlvalogatonak függvénynek kellene lennie, hogy visszaszólhasson a hívójának, vajon talált-e hasznos dokumentumot, vagy sem. Vázoljuk fel, hogyan is kellene kinéznie egy ilyen függvénynek:

```
function fajlvalogato(oFolder, szint)
  if van hasznos adat then
    fajlvalogato=True
  else
    fajlvalogato=False
  end if
end function
```

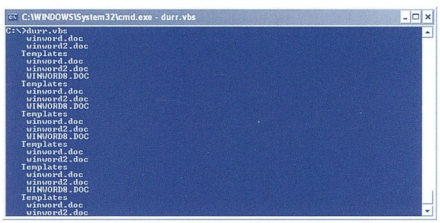
Ha találunk hasznos adatot, térjen vissza True *(igaz)* értékkel, ellenkező esetben pedig False *(hamis)* legyen a válasza.
Egyetlen dolgot nem határoztunk még meg: mitől hasznos egy adatfájl? Első közelítésben mondjuk akkor hasznos, ha a kiterjesztése .DOC. A *.DOC kiválogató függvény így nézne ki:

```
function fajlvalogato(oFolder, szint)
dim oFile
fajlvalogato=False
for each oFile in oFolder.Files
  if lcase(right(oFile.Name, 3)) = "doc" then
    REM *** HASZNOS ADAT! ***
    wscript.echo space(szint+1) & oFile.Name
    fajlvalogato=True
  end if
next
end function
```

Ha ezt meghívjuk a konyvtarlistazo eljárás legelső sorában úgy, hogy csak akkor írja ki a könyvtárneveket, ha azokban docot is talált…

```
if (fajlvalogato(oFolder, szint)) then
  wscript.echo space(szint) & oFolder.Name
end if
```

máris intelligensebb kódunk van, mint a DIR parancs. Mindössze annyi benne a „baki", hogy előbb írja ki a fájlneveket, és csak azután a könyvtárat, de kit zavar? És még csak a 0.97-es verziónál járunk!
Lefuttatva a legújabb verziót, meglepetéssel tapasztalhatjuk, hogy winword.doc mindenhol van:



Hová lett az üres hely a lemezen? Megette a winword.doc!

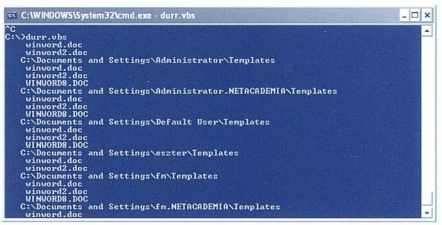
Sajnos a jelenlegi kód nem írja ki a könyvtárak teljes elérési útvonalát, így sötétben tapogatózunk, hol is tanyázhat ez a sok winword.doc, de szerencsére a Folder objektumnak van egy Path tagváltozója is. Ha kicseréljük az eredeti könyvtárnév-kiíró sort

```
8.    wscript.echo space(szint) & oFolder.Name
```

erre:

```
8.    wscript.echo space(szint) & oFolder.Path
```

mindjárt láthatjuk is, hol bujkál ez a rengeteg „hasznos" adat. Szanaszét a profilkönyvtárakban!



Találtunk néhány winword.doc fájlt a lemezen…

A következő feladat a hasznos adatok halmazának meghatározása.

DURR.VBS 0.98 – a felesleges elemek kihajítása

Mint látható, és a cikk elején levezetett gondolatmenetből is kiviláglik, kivételeket kell keresnünk, bizonyos dokumentumokat ki kell zárnunk a hasznos adatok közül.
Ehhez jó lenne felsorolni a kivételeket, és valami energiatakarékos módszerrel összehasonlítani az összes elemet a kivételek listájával. Szinte erre a feladatra találták ki a Dictionary *(szótár)* objektumot, ami nem más, mint egy szupertömb *(array)*. Annyiból szuper, hogy tudja, milyen elemek vannak benne, és kérdéseinkre válaszol is, tehát egymaga elvégzi a kivétellistával való összevetés nehéz feladatát.
Kicsit túlságosan is okos, mert nem pusztán értékeket, hanem értékpárosokat nyel le, amelyek közül az első a keresett kifejezés *(szó)*, a másik pedig a hozzá tartozó szócikk. Általában úgy használjuk a szótárakat, hogy a keresett szónál elolvassuk a szócikket. Nekünk azonban bőven elegendő szolgáltatás egy szó megtalálása, szócikként valami random karakterkupacot fogunk megadni.
Nézzünk egy háromsoros példát a használatára!

```
set szotar=createobject("scripting.dictionary")
szotar.add "alma", "az alma egy gyumolcs"
szotar.add "banan","a banan is egy gyumolcs"
if szotar.Exists("alma") then
  msgbox szotar.item("alma")
```

Ez a kis programocska a kételemű szótárból az "alma" szót keresi, és ha van ilyen, kiírja a hozzátartozó szócikket:

Ezt a fantasztikusan egyszerű keresési lehetőséget fogjuk felhasználni a nyilvánvalóan értéktelen dokumentumok kiszűrésére. Például így töltethetnénk fel a szótárat a winword.doc-ok kiszűréséhez:

```
set oSzotar=createobject("scripting.dictionary")
oSzotar.CompareMode = vbTextCompare
oSzotar.add "winword.doc", "az alma egy gyumolcs"
oSzotar.add "winword2.doc", "az alma egy gyumolcs"
oSzotar.add "winword8.doc", "az alma egy gyumolcs"
```

A szótárat létrehozása után, de annak üres állapotában a második sorban átállítom kisnagybetű-érzéketlen működésre *(case insensitive)*, hogy ne kelljen a fájlneveket állandóan kis- vagy nagybetűs formára hozni.
Figyeljük meg, hogy mivel a szócikkre valójában nincs szükségünk, oda tetszőleges szöveg kerülhet *(választhattam volna rövidebbet is...)*
A szótár feltöltését egyszer, a program elején kell elvégezni, a fájlok feldolgozásánál pedig már csak használjuk. Kicsit kibővítettem az eredeti if utasítást, hogy magában foglalja a szótárazást is:

```
if lcase(right(oFile.Name, 3)) = "doc" and
↳ not(oSzotar.Exists(oFile.Name)) then
```

DURR:VBS v0.99 – a szótár feltöltése fájlból

Egy kis fantáziával a szótár átalakítható úgy is, hogy egy textfájlból olvassa fel a saját aktuális tartalmát, így nem kell a scriptbe bedrótozni a kiszűrendő értékeket. Ehhez szintén a FileSystemObject fog segítséget nyújtani, mégpedig OpenTextFile és ReadLine metódusaival, a következőképpen:

```
set oLista = oFSO.OpenTextFile("kivetel.txt")
Do While oLista.AtEndOfStream <> True
  oSzotar.Add oLista.ReadLine, "Alma Ata"
Loop
oLista.Close
```

Az első sorban megnyitjuk a kivetel.txt nevű fájlt. A vállalkozóbb kedvűek némi hibakezeléssel is körbeágyazhatják. Ez ebben a formában például olyankor száll el, ha nincs kivetel.txt nevű fájl. Hát legyen! Én már nagyon unom a hibakezelést!
Ezután egy ciklussal soronként kiolvassuk belőle a tartalmat, és feltöltjük vele a szótárunkat, végül, mivel az illendőség úgy kívánja, lezárjuk a fájlt. Az „Alma Ata" ne zavarjon senkit, az az általunk semmire sem használt szócikk feltöltésére kiagyalt string, semmi több. És lassan eljutunk az 1.0-s verzióhoz. Ehhez már csak annyit kell hozzátenni, hogy a helyesen megtalált fáljokat a script mentse le a D:\ meghajtóra, és már készen is vagyunk.

DURR.VBS v1.0 – a megtalált adatok mentése

Mentésre használhatnánk a FileSystemObjectben található Copy metódust is, viszont kényelmetlen lenne puszta kézzel, kódból létrehozni ugyanazt a könyvtárszerkezetet, mint amiben a hasznos adatot találtuk. Legyünk ez alkalommal lusták és megalkuvók, és használjuk fel az XCOPY parancsot, ami pontosan úgy másol, ahogy arra nekünk szükségünk van.
Sőt, ha már XCOPY, ne is azonnal hajtsuk végre, hanem inkább írjunk a scriptünkkel egy batch fájlt, amiben sorakoznak az XCOPY-k. Ennek az az előnye is megvan, hogy a DURR.VBS futása után még átnézhetjük, mit is talált fontosnak, és ha kell, kitörölhetünk egy-két felesleges sort. Persze ezt az utólagos ellenőrzést nem lehet mind a 800 gépen megtenni, de a fontosabb helyeken *(pl. Vezér Elvtárs gépe)* igen.
A batchfájlt a script elején kellene megnyitni, és minden egyes hasznos adat megtalálásakor bele kell írni egy újabb XCOPY sort.

```
set oBatch = oFSO.CreateTextFile("mentes.bat")
```

Ez utóbbi kódot oda kell felvenni, ahol megjegyzésben már szerepel, hogy hasznos adatra leltünk, tehát ide:

```
REM *** HASZNOS ADAT! ***
  oBatch.WriteLine "XCOPY " oFolder.Path & "\"
  ↳ & oFile.Name & "D:\ /s"
```

És valahol a kód legvégén, amikor már visszatértünk a rekurzióból is, le kell zárnunk az oBatch fájlt:

```
oBatch.Close
```

Összegzés

Ha valakire olyan feladatot sóznak, amit száz mérnökév alatt lehet elvégezni, gondoljon szeretettel a scriptelési lehetőségekre, legyen az .NET konzolalkalmazás vagy akár Windows Scripting Host-kód. Mindkettőnek megvan a maga előnye és hátránya, de egy biztos: csak kóddal sokszorozhatjuk meg önmagunkat. Smith ügynök a Matrix2-ben szintén kóddal sokszorozta meg önmagát, tanuljunk tőle!
A DURR.VBS teljes forráskódját a technet weben helyeztem el, aki hibát talál benne, kérem jelentse. Nem szégyellem hisz 1.0-s változat. Más gyártók szoftvereinek tizenvalahanyadik változata is bugos!

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

A DURR.VBS 1.0 innen tölthető le:
[1] http://technet.netacademia.net/download

Kapcsolódó tanfolyamaink:
2433 – VBScript a gyakorlatban

Rendszerfelügyelet nagyvállalati környezetben

Avagy a dromedár szemfüle

Ahogyan az informatika egyre nagyobb teret követel magának a vállalati környezetben, úgy egyre komplexebbé válnak a rendszerek, egyre nagyobb rendelkezésre állást követelnek meg tőle, így egyre nagyobb az igény a rendszer állapotának nyomonkövetésére. A rendszerfelügyelet itt nyújt hathatós segítséget az üzemeltetés részére.

Jöttem, láttam, rendszer?

Amikor az új rendszergazda munkába áll, a lehető legritkább esetben kerül egy olyan helyre, ahol éppen most akarnak számítástechnikai rendszert bevezetni. A zöldmezős nagy projektek lassan, de biztosan a ködös múltba vesznek. Helyettük a mit is örököltem téma népszerű.

Kis rendszer – nagy szolgáltatáslista

Már egy szerényebb informatikai rendszer is sokféle szolgáltatást nyújt. A nagyobb pedig ennek minősített esete. Gondoljuk át, mi minden található egy átlagos informatikai rendszerben. Szinte minden rendszer része az irodai csomag:

- szövegszerkesztés
- táblázatkezelés
- elektronikus levelezés, fax
- böngészés *(igen, az Interneten!)*
- naptár- és névjegykezelés, kapcsolattartás

Ez bővül személyre szabott alkalmazásokkal:

- Ügyviteli rendszer *(Személyügy, pénzügy, eszközgazdálkodás, stb.)*
- Egyéb célalkalmazások *(Saját fejlesztések, illetve egyéb rendszerek: belső nyilvántartások, elszámolások, illetve egyéb munkafolyamatokat támogató alkalmazások.)*

Ezekhez a rendszerszolgáltatásokhoz egy általános informatikai hátteret is biztosítani kell, többek között a kiszolgálókkal, szerveroldali alkalmazásokkal:

- kiszolgáló operációs rendszer *(alapfeladatok)*
- irodai szerveralkalmazások
- adatbáziskezelés
- távoli hozzáférés
- ügyviteli rendszer

Ha már így együtt vannak, akkor ezt meg is kell*(ene)* védeni, ha jó az ellen:

- vírusvédelem
- tűzfal

Ami nem hiányozhat egyetlen rendszerből sem:

- mentés
- archiválás

A felhasználószám növekszik, egyre komplexebb, egymással kapcsolatban álló alkalmazások kerülnek be a rendszerbe, egyre komplexebb igényeket támasztanak a rendszerrel szemben. *(7x24-es üzem, 9x.z% rendelkezésre állás, stb.)* Egy szép napon nagyvállalati környezetté növi ki magát.

Az üzemeltetési tapasztalatok szerint a rendszer megbízhatósága, sebezhetősége az alábbi körülményektől erősen függ:

- A FELHASZNÁLÓK HASZNÁLJÁK!
- A hackerek feltörik!
- A vírusok elözönlik!
- A mentés behal!
- Az adatbázis elszáll!
- Eljő a kék halál!
- A diszk beadja a kulcsot!
- Az egész géptermet elnyeli a tsunami!

Ezeknek a rendszereseményeknek valahol nyomuk is marad:

- Eseménynapló *(Event log, ebből mindjárt legalább 3 is van egy Windowst futtató gépen)*
- Alkalmazásnapló *(minden alkalmazás szorgosan jegyzetelget, akár sok-sok külön naplóba is.)*
- Forgalmi napló *(A Webkiszolgáló, a tűzfal és társai írogatják.)*
- Teljesítménynapló
- Vírusvédelmi rendszer írásai *(Érdekes olvasmány!)*
- Egyéb <nagyonsajátfontos> rendszer állományai

Ezek mérete már egy közepesen fejlett rendszer esetén is egyre inkább túlmutat a Háború és béke terjedelmén. *(Több tízezer soros, megabájtos szöveges naplók képződnek naponta.)* Teljes átolvasásuk egyre kevésbé képzelhető el, elsősorban a lassú humán periféria korlátai miatt. *(A régi gondolatra, hogy majd csak a piros részekre koncentrálunk, a válasz: „Nem minden hiba piros, és nem minden piros hiba", aztán meg a szöveges naplókat nem is színezik.)*
A rendszerüzemeltetés munkája – ahol lelkiismeretesen végzik – itt az egekig ér.
Olyan ez, mint amikor Gizi néni rohan a kis falusi állomáson a biciklit tolva a vonat után, de nincs ideje felülni a biciklire, mert akkor lekési a vonatot.

Értesítések

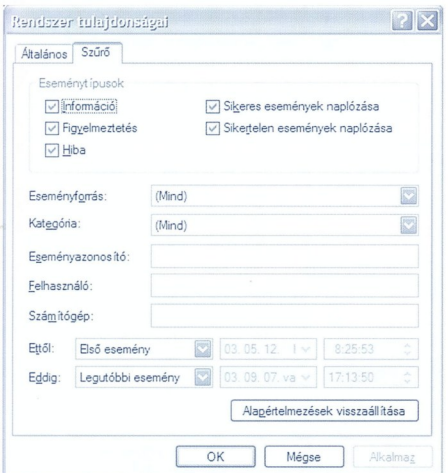
A rendszer különféle komponensei állapotukról néha üzenetet is tudnak küldeni *(pl.: Exchange, IIS stb.)*, de ez sem olyan

nagy öröm. Mi lenne, ha üzengetés helyett inkább meggyógyítaná önmagát?

Komponensek felügyelete

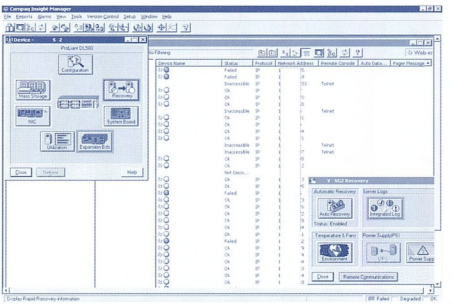
A napi felügyeleti kézimunka megkönnyítésére már eddig is léteztek különféle rendszerek, melyekkel egy-egy részfeladatot megoldhattunk:

- Eseménynapló szűrése, keresés
  Az eseménynaplóban minden feljegyzést megtalálhatunk – amit a rendszer jónak látott közölni velünk. Ezekben nem is olyan nagyon könnyű eligazodni, még az eseménynapló szűrő és keresési lehetőségeivel sem.



Az eseménynapló szűrő és keresési lehetőségei

- Beépített teljesítménymonitorozó eszközök: Perfmon: Az operációs rendszer, illetve számos alkalmazás különféle paraméterei figyelhetők meg akár élőben, vagy bizonyos időszak eseményei menthetők el későbbi kiértékelésre.
- A Notepad
  Elsősorban a szöveges naplóállományok megtekintésére használható. Különösen problémás a nagy állományok *(pl.: ISA log)* kezelése, nézegetése.
- A szerverhardverhez adott monitorozó program *(pl.: Insight manager, a HP Proliant kiszolgálók rendszermonitorozó programja, ahol a kiszolgálók állapota követhető nyomon, illetve sok egyéb információ is lekérhető.)*



Az Insight Manager képernyői

Ezek közös hibái:

- ☹ csak részegységeket látnak *(pl.: hardver)*
- ☹ az egyes alkalmazásokat, kiszolgálókat külön-külön kell beállítani, nyomon követni
- ☹ különféle prezentációs réteg *(install, beállítások)* a különféle alkalmazások különféleképpen jelenítik meg, szervezik csoportba az adatokat, ezeket más-más logika szerint lehet feldolgozni:
  - Naplóállomány: egyszerűen soronként vannak benne az adatok.
  - Eseménynapló: saját formátumú állományban tartalmazza az adatokat
- ☹ külön adatbázisok -> összetett lekérdezések ??? Ha többféle adat összefüggésére vagyunk kíváncsiak *(pl.: milyen processzorterhelés volt éppen akkor [perfmon], amikor az alkalmazás hibát jelzett [eseménynapló]?)* akkor ezt kézzel lekérdezni a különféle helyeken, illetve az adatkonverziókat – ha egyáltalán lehetséges – elvégezni, az eredményeket egymáshoz illeszteni – ha egyáltalán elvégezhető – nagyon nehéz munka.
- ☹ összetett hibák esetén több helyről is érkezik jelzés pl.: ha meghibásodik egy hálózati adapter, akkor sok, egymással első látásra nem összefüggő hibaüzenetet kaphatunk, ráadásul különféle formában: a rendszernaplóban eszközhiba; az Insight managerben hálózati eszközhiba; az alkalmazás naplójában: nem éri el az adatbázist hiba; stb.

A gordiuszi csomó

Minél több komponensből áll egy rendszer, annál több állapotmutatója létezik, sokféle működési jelenséget produkál. Egyre fontosabb minél jobban, folyamatosan figyelemmel kísérni működését, hogy megfelelően, gyorsan, reaktívan lehessen reagálni, elhárítani a működési rendellenességeket Egyre fontosabb proaktívan – már a hibajelenség fellépése előtt – megelőző lépéseket tenni a folyamatos zavartalan működés fenntartására.

Ezekhez a feladatokhoz kellene egy eszköz, amely:

- Központosított
- Önálló egységekből áll
- Megbízható
- Testreszabható
- Bővíthető
- Biztonságos

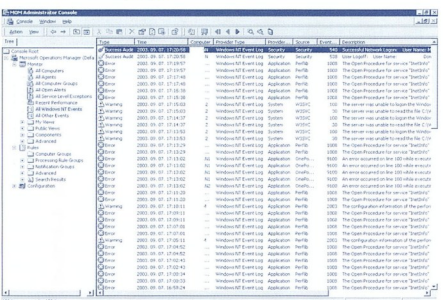
Ez a rendszerfelügyeleti eszköz.

A Microsoft termékpalettáján szerepel egy ilyen termék:

A Microsoft Operations Manager *(MoM)*.

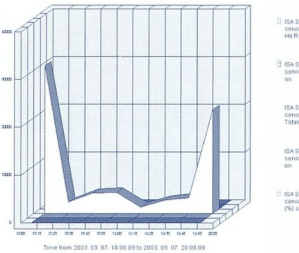
A MoM-ot a NetIQ fejlesztette, majd a Microsoft termékpalettájára került. Egy olyan rendszerfelügyeleti termék, amely sokféle szolgáltatást kínál:

- Könnyű telepíthetőség *(szerveroldalon beszédes telepítő, a kliensek automatikusan települnek)*
- Központi, megbízható adatbázis: MS SQL.
- Az igényeknek megfelelően skálázható *(Akár egyetlen MoM szerverrel is elindulhatunk, amit később az igényeknek megfelelően tovább bővíthetünk, sőt több MoM-ot is összekapcsolhatunk.)*
- Beépített, házi bejegyzésekkel bővíthető tudásbázis, amellyel a begyűjtött információkat értelmezhetjük, a hibaelhárításhoz hasznos segítséget nyújt, a helyi lehetőségek figyelembe vételével.
- Egyszerűen kezelhető MMC beépülő modullal, illetve webes konzol segítségével.
- Nyitott API *(más gyártók is bővíthetik csomagokkal.)*
- Biztonságos *(a kliensekkel titkosított csatornán kommunikál)*
- Saját, önálló kliensekkel rendelkezik
  - Automatikus települ
  - Szervízként fut
- A kliens képes összegyűjteni a felügyelt számítógépről adatokat:
  - Eseménynaplók



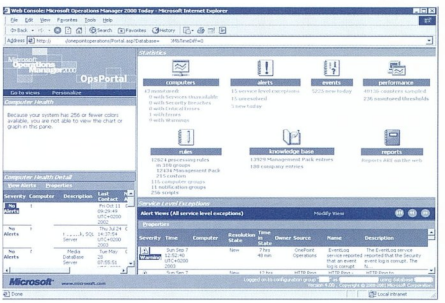
Összesített NT-eseménynaplók

  - Támogatott alkalmazások naplóállományai
  - Bármilyen más naplóállomány adatai, amennyiben meg tudjuk fogalmazni, hogy milyen módon épül fel.
- Ezeket az adatokat már a kliens képes megadott szabályok szerint szűrni.
- A szerveren további szűrések állíthatók be.
- A kliens képes az eseményeket konszolidálni, így röviden, egyszerű formában kerülnek a szerverhez. *(pl.: egy komplex hiba csak egy eseményként kerül jelentésre a szerver felé.)*
- Képes automatikus beavatkozásra, szabályok alapján automatikus cselekvési tervek megvalósítására. *(Programfuttatás, szerviz, kiszolgáló újraindítás, stb.)*
- Riasztások kiküldésére *(SMTP levelek)*
- A szerveren a kapott adatok alapján sokféle jelentést készíthetünk el rövid idő alatt. *(Komplex lekérdezések írhatók.)*



MoM jelentés

- Ezek akár a webes felületen is könnyen megjelenítehetők



MoM OpsPortal (webelérés)

Milyen rendszerkomponenseket támogat?

Röviden, a teljesség igénye nélkül a támogatott alkalmazások listája:

- Active Directory
- Internet Information Server
- Exchange 2000
- SQL 2000
- Terminal services
- DHCP
- DNS
- Systems Management Server
- WINS
- RRAS
- NT eseménynapló
- WMI események
- SNMP traps
- Syslog
- alkalmazásnaplók
- stb…

A második részben szó lesz a MoM felépítéséről, bővítőcsomagjairól, a tervezés és a bevezetés nehézségeiről, az üzemeltetéséről, illetve adok néhány hozzá kapcsolódó ötletet, tippet, melyek a mindennapi használatát könnyíthetik meg

Megyesi Barnabás
Megyesi.Barnabas@meh.hu
üzemeltetés vezető

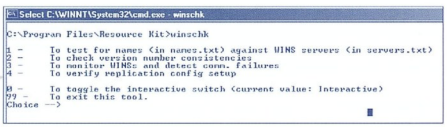
Gyógyszeresdoboz

Windows 2000 Server Resource Kit V.

Folytassuk tovább kisebb-nagyobb hálózati segédprogramokkal.

Winschk.exe

Ez a programocska főleg a WINS adatbázis konzisztencia vizsgálatára illetve a WINS szerverek között replikáció ellenőrzésére használható, de ezeken kívül is tud még egy-két hasznos dolgot. Természetesen a műveleteket parancssorból végezhetjük el és – ahogy az már megszokott a Resource Kit programoknál – akár távolból is. Menüvezérelt, az alábbi funkciókból választhatunk:



A WinsChk lehetőségei

Kiemelném az első pontot, amellyel ellenőrizhetjük az összes helyi WINS szerver adatbázisát, pl., abból az okból, hogy vajon a *(régebbi)* kliensek számára döntő fontosságú NetBIOS nevek *(pl. a DOMAIN[1C]) rekord, amellyel a tartományvezérlőket jelöli a WINS)* benne vannak-e ezekben az adatbázisokban? Az ellenőrzéshez előzetesen két szövegállományt kell megetetnünk vele: egy servers.txt nevűt, amelyik a hálózatunkban működő WINS szerver*(ek)* IP címét tartalmazza, valamint names.txt nevűt, amelyben egy-egy sor a keresendő neveket és rekordtípusokat tartalmazza csillaggal elválasztva és csupa nagybetűvel, pl. így:

```
TESTDOMAIN*1C
TESTDOMAIN*1B
```

Ha ez van bejegyezve a WinsChk.exe-vel megegyező mappában lévő names.txt-ben, akkor a tartomány tartományvezérlőinek illetve főtallózójának bejegyzéseit ellenőrizhetjük majd a WinsChk 1. pontja alatt. A csillag utáni bejegyzéseknek *(azaz a NetBIOS nevek 16. karakterének = NetBIOS suffix)* itt nézhetünk utána **[Q163409]**.
A kettes pont alól történhet meg a WINS Server*(ek)* konzisztenciaellenőrzése, a harmadikban pedig a replikáció vizsgálata, ami lehet folyamatos *(alapbeállítás szerint 3 óránként egyszer, de ez is állítható ugyanitt)*, vagy egyszeri is. Az eredmény pedig a monitor.log nevű állományban kerül naplózásra. A négyes pontban a replikációs partner*(ek)* beállításait vizsgálja a program.

WINS Replication Network Monitor Parser

A Network Monitor a jól képzett rendszergazda ezüstpuskája: ha már minden általános hibakeresés és KB/MSDN/Google tallózáson túl vagyunk, akkor elő lehet húzni a nyeregkápa alól. De sajnos a hálózati problémák nem gyáva indiánok, ettől még nem ijednek meg. Így aztán tanuljunk **[netmoncikk]** **[usingnetmon]**, de segítsünk is magunkon. Használjunk a szemünk előtt pörgő bitsorozatok ismert szekvenciáinak felismeréséhez *(a beépítetteken kívül)* külső protokoll parsereket, körülbelül úgy, mint a plug-in-okat a böngészőknél. Merthogy aránylag minimális, de manuális munkával lehetőség van utólag behegeszteni ezeket a NetMonba. A következő lépésekben a wins.dll azaz a WINS Replication Network Monitor Parser kerül beállításra, miután persze bemásoltuk a System32\Netmon\Parsers mappába:

1. Nyissuk meg a Tcpip.ini-t a System32\Netmon\Parsers mappából.
2. A **[TCP_HandoffSet]** részhez adjuk hozzá a "42 = WINS" bejegyzést, ahogyan itt is látható:

```
[TCP_HandoffSet]
20    = FTP
21    = FTP
23    = TELNET
25    = SMTP
42    = WINS
53    = DNS
```

3. Mentsük el és zárjuk be a Tcpip.ini-t.
4. Nyissuk meg a Parser.ini-t a System32\Netmon mappából.
5. A példához hasonlóan adjuk hozzá "WINS.DLL = 0: WINS" bejegyzést a [PARSERS] részhez.

```
[PARSERS]
FRAME.DLL    = 0: FRAME
MAC.DLL      = 0: ETHERNET, TOKENRING, FDDI, TMAC, SMT
LLC.DLL      = 0: LLC, RPL, SNAP, BPDU
NETBIOS.DLL = 0: NETBIOS
SMB.DLL      = 0: SMB
.....
WINS.DLL     = 0: WINS
```

6. A file végére írjuk be a következő sorokat, majd mentsük el és zárjuk be.

```
[WINS]
Comment      = "WINS Server Replication"
FollowSet    =
HelpFile     =
```

Persze ha van hozzá elég lelkierőnk, mi is írhatunk parsert **[nmparser]**. A Resource Kit kettő ilyet tartalmaz, a WINS replikációs parseren kívül a Windows Load Balancing Serverhez is található két .dll a CD-n. Persze a webről is be lehet egyet-kettőt szerezni *(pl. az NTP-hez **[sntpparser]**)*, de sajnos a sokat emlegetett Kerberos parser továbbra sem hozzáférhető. Még egy fontos tudnivalóról szólnom kell: az eredeti Resource Kitben lévő wins.dll hibás, a Microsoft később kiadott egy javított példányt, amelyet erről a címről **[reskitftp]** letölthetünk.

DNS Provider

Ezt az összetett csomagot a Resource Kit szabadon letölthető részben találjuk meg **[reskitftp]**. Két fő részből áll: a DNS

Windows Management Interface *(WMI)* Provider eszközből, amelyhez tartozó szkriptek VBScript-ben íródtak, és egy Perl szkript gyűjteményből. Mindkettő segítségével készíthetünk, szerkeszthetünk és törölhetünk DNS zónákat, hostokat, parancsorból és távoli gépen is, sőt az WMI változat Excel állományokból is képes zónákat, rekordokat, hostokat tehát egy komplett DNS alrendszert létrehozni. A letölthető dnsprov.zip ennek megfelelően tartalmaz egy DNSImport.vbs nevű szkriptet és egy Excel mintaállományt is. A telepítés menete a következő:

1. Másoljuk be a Dnsschema.mof és a Dnsprov.dll állományokat a \system32\wbem mappába.
2. Ugyanitt „hangoljuk rá" az eszközt a DNS szervizre a következő paranccsal:

```
mofcomp dnsschema.mof
```

3. Ezután már csak regisztrálni kell a Dnsprov.dll-t.

```
RegSvr32 dnsprov.dll
```

Ezt a műveletsort minden olyan szerveren meg kell ismételnünk, amelyen használni akarjuk az eszközt. Az alábbi szkripteket használhatjuk ezután:

Dnsimport.vbs	Zónák, hostok, rekordok készítése importálással az említett módon, Excelből
Dnsrecord.vbs	Hostok készítése parancssorból
Dnsserver.vbs	A DNS szerver paramétereinek beállítása
Dnszones.vbs	Zónák készítése

Így készítünk pl. egy primary dns zónát parancssorból:

```
cscript dnszones.vbs create primary forward
↳ altgr.hu
```

És lőn egy zónánk, de tegyünk bele egy A rekordot is:

```
cscript dnsrecord.vbs /add A altgr.hu srv1
↳ 192.168.1.221
```

Ez utóbbi művelet egy másik gépre, egy másik, már létező zónába:

```
cscript dnsrecord.vbs /add A ctrl.hu srv3
↳ 10.0.1.201 /S srv2 /U Administrator /W password
```

Ugye milyen egyszerű? A további DNS hekkelés céljából felhívnám a kedves Olvasó figyelmét a - szintén a csomagban lévő - dnsprovider.chm súgóállományra, amelybe elképesztő mennyiségű információt, leírást és példát zsúfoltak bele.

Getmac.exe

A Getmac.exe a hálózati interfészek MAC címeit valamint protokolljainak listáját adja vissza és NetBIOS illetve DNS nevekkel is használható. Nagyon sok helyen tudjuk használni pl. az előbb említett Network Monitornál is, az alapból MAC címek barátságosabb elnevezésének apropóján. A Windows Server 2003-ban is megtalálható egy alaposan felturbózott változata.

NetSet.exe

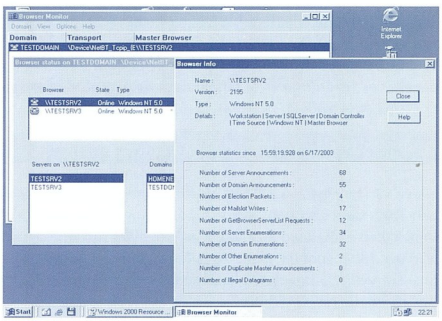
Ezzel az eszközzel a hálózati komponenseket konfigurálhatjuk, illetve telepíthetjük vagy leszedhetjük. Például sok azonos típusú számítógép esetén akár egy válaszfájl segítségével felügyelet nélküli telepítést, eltávolítást vagy konfiguráció visszaállítást is végezhetünk. A **/display** kapcsolóval pedig listát készíthetünk az aktuális hálózati ügyfelekről, szervizekről és protokollokról.

Wntlpcfg.exe

Találós kérdés: melyik az a Microsoft operációs rendszer, amelyikben parancssori és a grafikus felületen is megtekinthetjük a jelenlegi IP konfigurációnkat? Nem, nem az XP, és nem is a Windows Server 2003. Hanem a Windows 98 ☺. Bizony, itt van **ipconfig** és van **winipcfg** is. A többin vagy az egyik, vagy a másik. De ha akarjuk *(vagy éppen megszoktuk)*, ezzel a programocskával a *(teljes NT platformon)* a GUI-n is láthatóvá válnak az IP beállítások.

Browmon.exe

Ki ne ismerné a nyolcezressel kezdődő Event Logban szereplő piros X-es táblákat, amelyek mind a túlságosan demokratikus *(értsd: szavazáson alapuló)* módszerrel működő Tallózó szolgátatás hibáira utalnak *(pl. 8003, a Master Browser anomália)*. A Browser Monitor segít a hibák felderítésben, képes megmutatni tartományonként pl. a főtallózót, annak egyéb funkcióit valamint alapos statisztikát is ad a kezünkbe a grafikus felületen.



Kellemes kis tallózási statisztika

Gál Tamás
MCSE 2000
gtamas@tjszki.hu

A Windows 2003 TCP/IP-implementációja

A TCP/IP-verem felépítése és főbb funkciói

A TCP/IP az Internet működésének alapja, és mára a vállalati hálózatokban is a legnépszerűbb hálózati protokollá vált. Útválasztási funkciói maximális rugalmasságot biztosítanak a nagyvállalati hálózatokban is. A TCP/IP-verem megvalósítása és szolgáltatásai tehát döntő fontosságúak minden operációs rendszer hálózati funkcióinak felépítésében.

A TCP/IP és a Windows operációs rendszerek

A Microsoft a 90-es évek elején indította el azt a projektet, amelynek célja egy olyan TCP/IP-verem létrehozása volt, amely nagymértékben megnöveli a Windows alapú hálózatok skálázhatóságát és teljesítményét. A teljesen újraírt, szabványos TCP/IP-implementáció a Microsoft Windows NT 3.5 operációs rendszerben mutatkozott be, és ezután minden egyes NT alapú Windows verzió megjelenésével továbbfejlődött, új képességeket, szolgáltatásokat vonultatott fel, és a teljesítmény növekedésével párhuzamosan egyre biztonságosabbá és megbízhatóbbá vált.

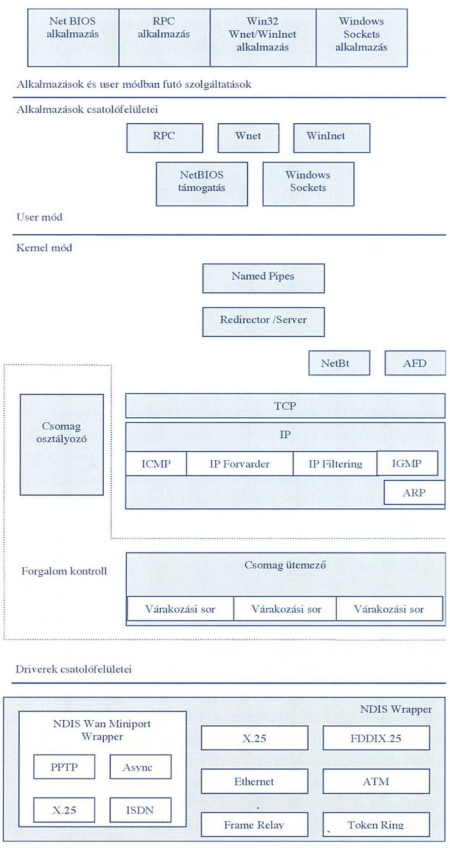
A TCP/IP-stack folyamatos fejlesztésének céljai szerint az implementáció:

- szabványos és minél szélesebb körű együttműködésre képes
- hordozható
- jól skálázható
- gyors
- rugalmas
- önbeállításra képes és könnyen felügyelhető.

A Windows Server 2003 TCP/IP-implementációja lehetővé teszi, hogy a Microsoft rendszereket nagyméretű vállalati, kormányzati, vagy nyilvános hálózatokba integráljuk, és megteremti az így felépített rendszerek biztonságos üzemeltetésének feltételeit.
A Windows Server 2003 telepítésekor a TCP/IP-protokoll is telepítésre kerül, és a korábbi verziókkal ellentétben eltávolítása sem lehetséges.

A Windows 2003 TCP/IP-architektúrája

A Windows TCP/IP központi elemekből *(például ARP, IP, TCP protokollok, stb.)*, szolgáltatásokból *(DHCP, DNS, stb.)* és a köztük lévő csatolófelületekből áll. A csatolófelületek közül a TDI *(Transport Driver Interface)* és az NDIS *(Network Driver Interface Specification)* nyilvánosan hozzáférhetők, az **[whdcddk]** címen elérhető Microsoft Windows Driver Development Kit-ek *(DDK)* tartalmazzák specifikációt és a programozási információkat.



A Windows 2003 TCP/IP szerkezeti modellje

Az NDIS-csatolófelület és az adatkapcsolati réteg *(data link layer)* funkciói

A Microsoft hálózati protokolljai a Network Device Driver Specification *(NDIS)* segítségével kommunikálnak a hálózati csatolók drivereivel. Az OSI-modell adatkapcsolati rétegének jelentős részét is a protokoll stack valósítja meg, hogy a hálózati csatolók drivereinek fejlesztése egyszerűbb lehessen. A Windows 2003 az NDIS 5.1-es verzióját tartalmazza, amelynek új funkciói a következők:

- **Értesítés Plug and Play és az energiagazdálkodással kapcsolatos eseményekről** *(Plug and Play and Power Event Notification)* – Lehetővé teszi, hogy a hálózati csatolók miniport driverei értesítést kapjanak új eszköz csatlakoztatásáról és eltávolításáról, vagy például a készenléti üzemmódból való visszatérésről.
- **Küldés leállításának támogatása *(Send Cancellation)*** – A hálózati protokolloknak nem kell hosszú ideig várniuk a küldési kérelmek teljesítésére.
- **Megnövelt statisztikai számlálók *(64 bit)*** – Ez a változás lehetővé teszi a korrekt hálózati statisztikák megjelenítését még nagy sebességű hálózati csatolások esetén is.
- **Teljesítménnyel kapcsolatos fejlesztések** – Számos változás történt annak érdekében, hogy a kritikus hálózati utak gyorsabbak lehessenek, és ne készüljenek fölösleges másolatok a hálózati csomagokról.
- **Változás a Wake on LAN szolgáltatásban** – Beállíthatjuk, hogy csak a magic packet-ek váltsanak ki ébresztést.

A Windows 2003 része továbbá a távoli NDIS *(Remote NDIS)*, amely külső szoftver telepítése nélkül lehetővé teszi az USB csatlakozású hálózati eszközök használatát.
Az adatkapcsolati réteg funkcióit a hálózati csatoló hardver a hozzá tartozó driver szoftver és a TCP/IP-stack alacsony szintű része valósítja meg. A hálózati csatoló által hardveresen megvalósított szűrők a csomagokban lévő MAC *(media access control)* alapján válogatják ki az adott gépnek érkezett csomagokat. A kártya a beérkező csomagok cél *(destination)* címmezőjét figyeli, és minden csomagot eldob, amelyben a mező értéke nem felel meg az alábbi feltételek valamelyikének *(vagy hibás CRC-vel érkezik)*:

- A mező a csatoló saját címét tartalmazza
- A mező broadcast címet tartalmaz *(0xFF-FF-FF-FF-FF-FF)*
- A mező olyan multicast címet tartalmaz, amelynek figyelésére a protokoll driver utasítást kapott az NDISRequest() függvény használatával.

A legtöbb hálózati csatoló esetében azonban ez a szűrés kikapcsolható, ebben az esetben minden csomag *(amelynek CRC-je megfelelő)* továbbkerül a meghajtóprogramhoz.
Mivel a szűrés hardveresen történik, a csomagok válogatása egyáltalán nem igényel CPU-időt. A szűrőn átjutott csomagokat ezután a hálózati csatoló meghajtóprogramja kapja meg hardver megszakítás segítségével. Mivel a meghajtóprogram az adott számítógépen futó szoftver, az idáig eljutó csomagok feldolgozása már CPU-időt is igényel. Ezután a meghajtóprogram a rendszermemóriába másolja a csomagot, és értesíti a csatolt szállítási meghajtókat.
Miközben a csomag a hálózaton vagy hálózatokon halad át, a benne lévő forrás *(source)* MAC-cím mindig annak a hálózati csatolónak a címe, amely a csomagot a médiára helyezte, a célcím *(destination address)* pedig azé, amelynek majd a csomagot le kell vennie a médiáról.
Ez azt jelenti tehát, hogy ha a csomag útválasztókon *(router vagy hálózati rétegben működő, Layer 3 switch)* halad át, a forrás és célcímek minden egyes ilyen eszközön áthaladva változni fognak.
Az adatkapcsolati réteg feladatai közé tartozik még az adott médiához tartozó Maximum Transmission Unit *(legnagyobb átvihető adategység, MTU)* meghatározása és az érték továbbítása a feljebb lévő protokollok felé. A csatolóhoz tartozó MTU-t felhasználja például a TCP-protokoll is az adott média optimális csomagméreteinek meghatározásához.

A központi komponensek és a TDI csatolófelület

A TCP/IP központi komponenseit a Windows Server 2003 tcpip.sys drivere valósítja meg. A komponensek a hálózati csatoló felől az NDIS, az alkalmazások felől pedig a TDI csatolófelületen keresztül érhetők el. A legfontosabb központi komponensek a következők:

- Address Resolution Protocol *(ARP)*
- Internet Protocol *(IP)*
- Internet Control Message Protocol *(ICMP)*
- Internet Protocol Security *(IPSec)*
- Internet Group Management Protocol *(IGMP)*
- Transmission Control Protocol *(TCP)*
- User Datagram Protocol *(UDP)*

Az **Address Resolution Protocol** *(címfeloldási protokoll)* végzi a kimenő csomagok MAC-címének meghatározását az IP-címek alapján. Minden kifelé tartó IP-datagramhoz hozzá kell csatolni a forrás és a cél MAC-címeit. A cél MAC-címének meghatározásához az ARP broadcast kérést küld ki az alhálózatra. Az ARP-kérelem megválaszolásakor mind a válasz küldője, mind az eredeti kérelmező feljegyzi a másik IP és MAC címét egy helyi táblázatba, melynek neve ARP-gyorsítótár *(cache)*.

Az Address Resolution Protocol működése

A gyorsítótárat az ARP a broadcast üzenetek számának csökkentésének érdekében tartja fenn. Ebben megtalálhatók az IP-címekhez tartozó MAC-címek. Az ARP-gyorsítótár mind statikus, mind dinamikus bejegyzéseket tartalmazhat. A dinamikus bejegyzések az idő előrehaladtával automatikusan kerülnek hozzáadásra és eltávolításra, a statikus bejegyzések viszont a számítógép újraindításáig a gyorsítótárban maradnak.
Az ARP-gyorsítótár dinamikus bejegyzéseinek várható élettartama 10 perc. A gyorsítótárhoz hozzáadott minden új bejegyzés időbélyeget kap, és ha a hozzáadástól számított 2 percen

belül nincsen rá újra szükség, érvényessége lejár, kikerül az ARP-gyorsítótárból. Az újra használt bejegyzések élete minden alkalommal két perccel meghosszabbodik, de 10 perc után mindenképpen törlődnek a gyorsítótárból. Az ARP-gyorsítótár tartalmát az

```
C:\arp -a
```

paranccsal jeleníthetjük meg.
Az ARP arra is felhasználható, hogy helyi útválasztóknak továbbítson IP-datagramokat, ha a csomag célállomása, nem az adott alhálózaton van. Ilyen esetben az ARP a helyi hálózaton működő útválasztó MAC-címét keresi meg.

Az **Internet Protocol *(IP)*** feladata a csomagok *(ezen a szinten a csomagok hivatalos neve datagram)* címzése és továbbítása. Minden IP adatcsomag tartalmazza a forrás és a cél IP-címét. A MAC-címektől eltérően a csomagban lévő IP-címek a datagram teljes élete alatt állandóak maradnak. Az IP-réteg alapfunkciói a következők:

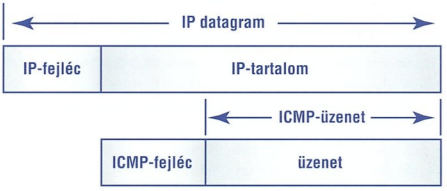
- **Útválasztás** – Az IP- réteg elsődleges funkciója az útválasztás. Az adatcsomagok az UDP, TCP, stb. protokolloktól, illetve a hálózati csatolóktól érkeznek az IP-hez. Az IP-fejlécet *(cél IP-cím)* megvizsgálva és az útvonaltáblával *(routing table)* összehasonlítva az IP eldönti, hogy az adott csomagot a felső rétegek vagy a hálózati csatoló felé kell továbbítania, esetleg egyszerűen eldobhatja azt. A Windows Server 2003 TCP/IP új lehetősége az **Automatikus metrika** opció megadása a hálózati csatoló tulajdonságlapján. A metrika az IP útválasztási táblázatban az adott útvonal költségét jelző szám. Lehetővé teszi a legjobb útvonal kiválasztását az adott célhoz vezető lehetséges útvonalak közül. Ha az opciót engedélyezzük, a TCP/IP automatikusan határozza meg az útválasztási táblázatban lévő útvonalak metrikáját az IP-cím, az alhálózati maszk és a helyi hálózati kapcsolatok alapértelmezett átjárója alapján. A kapcsolat metrikájának automatikus meghatározása, amely alapértelmezés szerint engedélyezve van, meghatározza minden egyes kapcsolat sebességét, és minden egyes kapcsolat útvonalának metrikáját úgy állítja be, hogy a leggyorsabb kapcsolat a legalacsonyabb metrikával hozza létre az útvonalakat.
- **IP cím ütközés érzékelése** – a duplikált IP-címek keresése a TCP/IP stack inicializálásakor, illetve új IP-cím beállításakor történik. Ekkor a gép broadcast ARP-üzenetek formájában kiküldi a beállított IP-címet a hálózatra. A kiküldött ARP kérések száma alapértelmezés szerint három, de a szám a

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
↳ Services:\Tcpip\Parameters\ArpRetryCount
```

registry érték létrehozásával beállítható.
Ha egy másik számítógép válaszol az ARP-kérések valamelyikére, az IP-cím már használatban van a hálózaton. Statikus IP-cím esetén az esetben a gép az adott hálózati csatoló letiltásával indul, hibaüzenet kerül a rendszernaplóba, és jelenik meg a képernyőn. Ha az IP-cím DHCP-kiszolgálótól származik, a gép nem tiltja le a csatolót, hanem DHCPDecline üzenetben értesíti a kiszolgálót a címütközésről, és új címet kér.

Internet Control Message Protocol *(ICMP)* – Az ICMP segítségével az IP-kommunikációt használó állomások és útválasztók hibákat jelezhetnek, és korlátozott adatcserét folytathatnak a vezérlésről és állapotokról.
Az ICMP-üzeneteket a Windows 2003 Server a következő feladatokra használja:

- Az útválasztó vagy célállomás kézbesítési problémáinak jelzése.
- Az útválasztó táblák felépítése és karbantartása.
- Útválasztó felderítés *(Router Discovery)*
- A Path Maximum Transmission Unit *(útvonal legnagyobb átviteli egysége, PMTU)* meghatározásában való közreműködés.
- Elérhetőségi problémák felderítése *(ping, tracert, pathping)*.
- Az IP-útválasztó *(router)* ICMP-üzenet formájában jelzi túlterheltségét a küldő állomásoknak.



IP datagramba ágyazott ICMP-üzenet

Internet Protocol Security *(IPSec)* – Az IPSec titkosításra épülő védelmi szolgáltatások és biztonsági protokollok együttese, biztosítja az adatintegritást, az adatok eredetének hitelesítését, az adatok titkosságát, és az adatáramlás titkosítását. Használata nem igényli az alkalmazások, illetve a protokollok módosítását, így az IPSec könnyen telepíthető meglévő hálózatokon is.
Az IPSec az alábbiakban felsorolt biztonsági szolgáltatásokat kínálja, illetve használja.

- Biztonsági tulajdonságok
- Nyilvános kulcsú tanúsítványon alapuló hitelesítés
- Előmegosztott kulcson alapuló hitelesítés
- Nyilvános kulcsú kriptográfia
- Az adatok sértetlenségének védelme kivonatoló függvényekkel
- Az adatok védelme titkosítással
- Kulcskezelés

Internet Group Management Protocol *(IGMP)* – Az IGMP olyan protokoll, amelynek segítségével az IP-állomások közlik a velük szomszédos csoportos küldést *(multicast)* alkalmazó útválasztókkal, hogy mely csoportos küldési csoportok tagjai. Az IP csoportos küldés *(multicast)* célállomások csoportjának történő adatcsomag küldést jelent. A multicast adatcsomag a célcsoport minden címére megérkezik, a unicast IP csomagküldéshez hasonló megbízhatósági szinttel, vagyis a datagram hibátlan megérkezése és a helyes sorrend nem garantált a csoport minden tagja számára.
A csoportok tagsága dinamikusan változik – az állomások bármikor csatlakozhatnak, vagy távozhatnak a csoportokból. A csoporttagok fizikai helyére vagy számára vonatkozóan semmilyen korlátozás nincs, és egy állomás több csoport tagja is lehet egyszerre.

Transmission Control Protocol *(TCP)* – A TCP összekötettés alapú, megbízható kapcsolatot biztosít az alkalmazások számára. A Microsoft rendszerek hálózatkezelése a TCP-protokollt használja a bejelentkezéshez, a fájlok és nyomtatók megosztásához, a tartományvezérlők közötti replikációhoz, a tallózó *(browse)* információk cseréjéhez, és más funkciókhoz. A Windows 2003 által támogatott főbb TCP-funkciók a következők:

- TCP vételi ablak méretének meghatározása és beállítása.
- Késleltetett visszaigazolás
- Szelektív visszaigazolás
- TCP időbélyegek
- Útvonalhoz tartozó legnagyobb átviteli egység *(Path Maximum Transmission Unit, PMTU)* meghatározása.
- Nem működő átjáró felismerése *(Dead Gateway detection)*
- TCP csomag újraküldési módok

A TCP-protokoll tervezésekor fontos szempont volt, hogy a legkülönbözőbb kapcsolati feltételek esetében is képes legyen az optimális átviteli teljesítményt biztosítani. Az adott kapcsolaton elérhető sávszélesség legfontosabb tényezői a következők:

- A kapcsolat sebessége *(másodpercenként átvihető bitek száma)*
- Továbbítási késleltetés
- Átviteli ablak mérete
- A kapcsolat megbízhatósága
- A hálózat és az átviteli eszközök terhelése
- Útvonalhoz tartozó legnagyobb átviteli egység

User Datagram Protocol *(UDP)* – Az UDP olyan, kapcsolat nélküli datagram-szolgáltatást biztosító TCP-kiegészítés, amely sem a kézbesítést, sem a csomagok megfelelő sorrendjét nem garantálja. A Microsoft hálózati komponensei bejelentkezéshez, tallózáshoz, és névfeloldáshoz használják az UDP-protokollt.

Transport Driver Interface *(TDI)* – A TDI olyan általános rutincsoport, amely lehetővé teszi, hogy a különböző hálózati rétegekben lévő összetevők kommunikáljanak az OSI-modell munkameneti rétegével. Ezen rutinok segítségével az átviteli réteg alatt és fölött elhelyezkedő programösszetevők a program módosítása nélkül működhetnek együtt.
A Microsoft azért fejlesztette ki a TDI-t, hogy a régebbi csatolófelületeknél *(NetBIOS, Windows Sockets)* nagyobb rugalmasságot, és bővebb funkcionalitást biztosítson. A TDI-specifikáció alapvető funkciókat ír le, amelyek segítségével a meghajtóprogramok és a TDI-ügyfelek kommunikálhatnak egymással.

A Windows 2003 Server redirector és server szolgáltatása közvetlenül A TDI-csatolófelületet használja a NetBIOS helyett, mivel így megkerülhető a NetBIOS számos korlátozása.

Hálózati alkalmazások csatolófelületei

A hálózati alkalmazások számos felületen keresztül kommunikálhatnak a TCP/IP központi komponenseivel:

- A redirector szolgáltatáson keresztül *(például named pipes)*
- A NetBIOS csatolófelület segítségével *(NetBIOS over TCP/IP)*
- A Windows Sockets csatolófelület segítségével

A legfontosabb ügyfélszolgáltatások

Az egyik legfontosabb ügyfélszolgáltatás a Dynamic Host Configuration Protocol *(DHCP)*. A DHCP a helyi hálózaton lévő DHCP-kiszolgáló IP-címekből álló adatbázisából teszi lehetővé az IP-címek dinamikus hozzárendelését az ügyfelekhez.
A Windows 2000 rendszerrel működő számítógépek alapértelmezés szerint először megpróbálnak a hálózaton kapcsolatot létesíteni egy DHCP-kiszolgálóval, hogy dinamikus beállítási adatokat kapjanak a telepített hálózati kapcsolatokról.
Ha a DHCP-kiszolgálót sikerült elérni, és a kiosztott beállítások működnek, akkor a TCP/IP beállítása sikeresen befejeződik.
Ha nem sikerült elérni a DHCP-kiszolgálót, akkor a számítógép az APIPA *(Automatic Private IP Addressing)* szolgáltatással automatikusan beállítja a TCP/IP protokollt. Ha az APIPA szolgáltatást használja, a Windows 2003 a 169.254.0.1 és 169.254.255.254 értékek közötti fenntartott IP-címtartományból jelöl ki címet. Ez az IP-cím a DHCP-kiszolgáló helyének meghatározásáig átmeneti beállításként fog működni. Az alhálózati maszk a 255.255.0.0 értéket kapja.
A Windows Server 2003 támogatja a DNS adatok dinamikus frissítését, a DHCP-kiszolgáló által kiosztott IP-címek is automatikusan bekerülnek a DNS-kiszolgáló adatbázisába.

Összegzés

A Windows 2003 TCP/IP implementációja számos szabványos funkció támogatását tartalmazza, a korábbiakhoz képest számos teljesítmény-optimalizáló változást, és új szolgáltatást tartalmaz. A Windows Server 2003 így kiválóan beilleszthető, és biztonságosan üzemeltethető komplex, nagyméretű hálózati környezetekben is.

Szerényi László
szerenyi.l@met.hu

A DHCP rejtett szépségei – II.

A címkiosztó szolgáltatás üzembehelyezésekor különösen fontos jól megválasztani a helyszínt. Egyszerű környezetben, ha nincs is több telephely, ez nem probléma, ma azonban már egyre több összetett hálózat működik. Lehet a topológia bármilyen bonyolult, a DHCP szervereknek megfelelően kell működniük.

Az egyszerű nagyszerű?

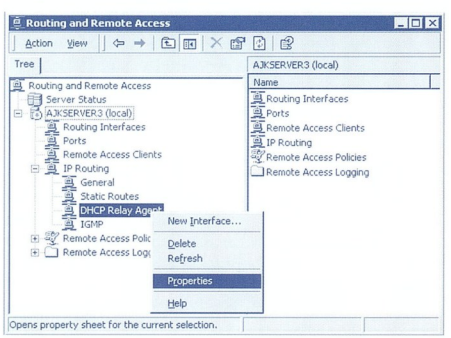
Tudjuk már, hogy a DHCP-rendszerek a szórt üzenetekre építenek. Nincs is más lehetőségük, hiszen a minden információáramlás alapját jelentő egyedi IP-cím kiosztása a feladatuk. Egy „kommunikációképtelen" ügyféllel kommunikálnak, s végül alkalmassá teszik őket az adatcserére.

A szórt üzenetek azonban a helyi hálózat foglyai, az útválasztók nem engedik át a broadcast csomagokat – ezáltal csökkentve a WAN vonalak és a többi LAN terhelését.

A fenti két állításból az következik, hogy minden DHCP-kiszolgáló a saját LAN-jának csapdájában kellene vergődnie, s bár helyben nagyszerű, távoli hatása nincs. A huszáros megoldás, amely a hálózat aktív tagjaivá teszi az állomásokat, csődöt mond, ha ki kellene terjeszteni. Hacsak nincs a tarsolyban egy újabb trükk. Nos, van ilyen.

A DHCP továbbító ügynökök *(Relay Agent)*

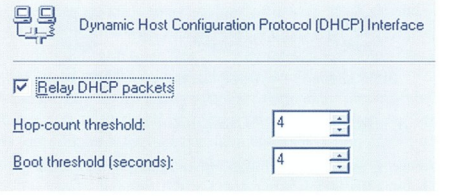
A Windows NT4, Windows 2000 és Windows 2003 rendszereknek van egy speciális, a DHCP-szervereket támogató szolgáltatása, ezek a továbbító ügynökök. Az NT4-ben még külön szolgáltatásként telepíthető *(gyakran összekeverték a DHCP kiszolgálóval és telepítették is)*, a Windows 2000-ben az RRAS szolgáltatás része.



A továbbító ügynök az RRAS szolgáltatás egyik funkciója

Egy továbbító ügynök feladata borzasztóan egyszerű: elkapja a helyi hálózatban kószáló DHCP-kiszolgálóknak szánt csomagokat, majd azokat a beállításai szerint egy vagy több – más alhálózaton működő – DHCP szerver felé továbbítja. Egyből érthető a trükk: a szórt üzenetek az ügynök „kezei közt" egyszerű unicast csomagokká válnak, így könnyedén, akár több útválasztón is áthaladhatnak. Mindezt azért tudja megtenni a Relay Agent, mert a DHCP szabvány, egészen pontosan az RFC 1542, a DHCP csomagban definiál egy „Relay IP Address" mezőt *(Ezt ismerik úgy is, mint „Gateway IP Address, vagy GIADDR)*. Az ügynök ide írja bele a saját címét, ettől egy csapásra minden DHCP kiszolgáló unicast csomagokkal kezd el kommunikálni vele. A DHCP szerver az ügynök kérését kiértékeli, méghozzá az IP-címe alapján. Azt vizsgálja, hogy az IP-címhez, amelyről a kérés érkezett, tartozik-e egy scope. Ha például az ügynök címe 10.1.0.10/16, van-e olyan bérlettartomány a szerveren, amelynek akár ez a cím is tagja lehetne. Ha igen, a szolgáltatás kiad egy bérletet. A megérkező válaszokat úgy küldi tovább a Relay Agent az ügyfeleknek, mintha csak egy valódi címkezelő lenne.

Az egyszerű működéséhez egyszerű konfigurálás tartozik. A beállítás csupán négy paramétert érint. Meg kell adnunk azt az interfészt *(hálózati kártyát)*, amelyre a szolgáltatás működni kezd, és fel kell sorolnunk azokat a DHCP-kiszolgálókat, amelyeket az ügynök értesít, ha helyi címkérelmet fog el.



Az ügynökkonfigurálás nem ördöngösség

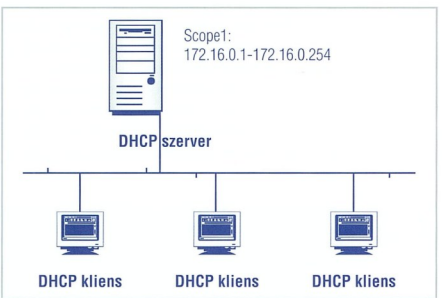
Ezen felül meg lehet változtatnunk az un. Hop-count threshold értéket. A paraméternek igazi jelentősége nincs. Tulajdonképpen az IP csomag TTL értékéhez hasonló mezőről van szó, de attól külön kezelik. Azt jelenti, hogy maximum ennyi Relay Agenten keresztül jöhet a csomag. Azért nincs a gyakorlatban jelentősége, mert bár a szabvány szerint egy ügynök az elkapott címet broadcast, multicast vagy unicast címre továbbíthatja, és így elméletileg elképzelhető, hogy több Agentnél is megfordul a DHCP-Discover csomag, gyakorlatilag kizárólag unicast címeket használnak az ügynökök, közvetlenül a kiszolgálókat megcímezve, tehát szinte sohasem kap ez a mező 1-nél magasabb értéket.

A Boot threshold paraméter már hasznosabb. Ennek segítségével két külön hálózatban lévő DHCP kiszolgálót *(egyéb konfigurációs trükkök mellett)* egymás tartalékává tehetjük méghozzá úgy, hogy a helyi kiszolgáló lesz a preferált. A boot thresholdban meghatározott másodpercig vár ugyanis az ügynök, mielőtt továbbítaná az üzenetet a *(távoli)* kiszolgálónak. Amennyiben van helyi DHCP szerverünk, és az ügynököt

csak arra használjuk, hogy a távoli kiszolgálót, mint tartalékot elérje, a négy másodperc bőven elegendő a helyi címkiosztónak az IP-cím allokálásához, tehát a kívánatos rendszert használták az ügyfelek. Amennyiben a helyi DHCP-kiszolgáló nem működik, az ügynök elvégzi a rábízott munkát és továbbítja a tartalék rendszer felé a kérelmeket. A DHCP rendelkezésre állásának növelését szolgáló technikák ismertetésekor erre a felépítésre még visszatérünk.

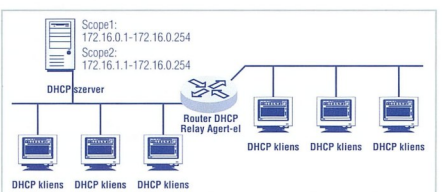
A hálózat-topológia és a DHCP

Most, hogy az ügynökökkel a címkiosztó szolgáltatás olyan, akár a „börtönéből szabadult sas", nézzük sorra a kiszolgálók elhelyezésének alapeseteit. Szeretném előrebocsátani, hogy az alábbiaknál sokkal bonyolultabb elrendezések is léteznek, amelyeket később majd tárgyalunk is, most csak a leglényegesebbeket tekintjük át.
A teljesen egyértelmű környezet, amikor nincs is útválasztónk, csupán egy LAN-unk, rajta ügyfelek és egyetlen DHCP-szerver. Elegendő egyetlen scope, habár korábban láthattuk, hogy nem lehetetlen több cím-intervallum használata sem.



A kályha: egy DHCP kiszolgáló, egy alhálózat

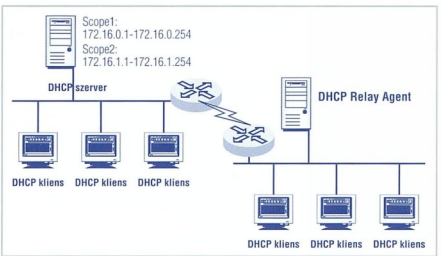
Az előző esetnél egy icipicit bonyolultabb, amikor több LAN-t egy útválasztó köt össze. Ekkor a távolabbi helyi hálózat ügyfelei csak akkor kaphatnak IP-címet, ha a router maga futtat egy DHCP továbbító ügynöki funkciót. Ma már az útválasztók többsége képes erre – csupán annyit kell tennünk, hogy a megfelelő beállításokat elvégezzük.



Egy címkiosztó szerver több alhálózattal

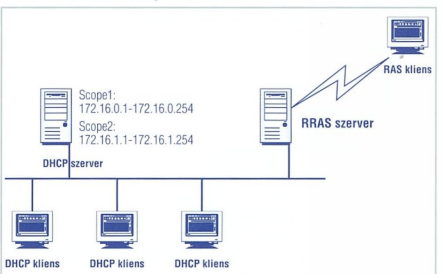
A fenti ábrán látható konfiguráció azonban sok esetben csak illúzió. Először is ritka eset, amikor egy helyi hálózatot egyetlen router választ el egy másiktól. Egyetemi hálózatokon, egy-egy nagyobb város hálózatában elképzelhető, ezzel együtt nem jellemző. Másrészt a topológia azt feltételezi, hogy az útválasztó konfigurálása a kisujjunkban van.

Kis magyar valóságunk nem ilyen. Ha vannak routerek, azt többnyire a külső szállító felügyeli *(sokszor ő is birtokolja)*. Ha a címkiosztást ilyen körülmények között a magunk kezében akarjuk tudni, úgy szükségünk lesz egy saját továbbító ügynökre.
Ekkor már mindegy, hogy hány eszköz választja el egymástól a helyi hálózatokat, s hogy azok milyen messze vannak egymástól. A következő ábra modellezi azt a szituációt, amelyben a leginkább elkél a továbbító ügynök. Könnyű elképzelni, hogy a bal oldali hálózatához, a központhoz, több olyan LAN kapcsolódik, mint a jobb oldalon látható, vagyis egyetlen DHCP szolgáltatást akár több tucat telephelyet is elláthat. Továbbító ügynökök kérdése az egész.



Ha nem mi felügyeljük az útválasztókat

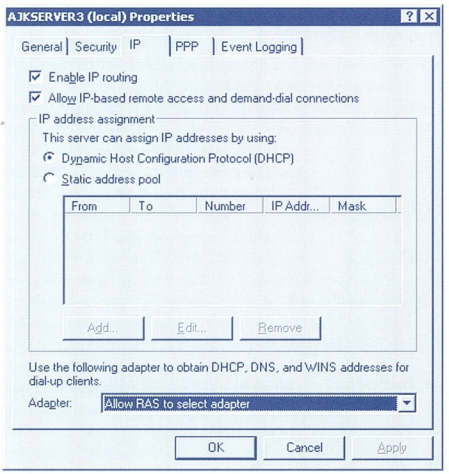
A skálázhatóságról csak annyit, hogy a Microsoft ajánlása szerint ne rakjunk egy DHCP-kiszolgálóra 1000-nél *(!)* több bérlettartományt. Összesen pedig ne legyen több, mint 10000 ügyfele egy szervernek. Ha valaki takarékoskodni szeretne a címkiosztó szerverekkel, akár a legnagyobb kiterjedésű magyar LAN is néhány szerverrel elüzemel. Ugye, hogy börtönéből szabadult a DHCP?
A továbbító ügynöknek még egy nagyon fontos szerep jut, ugyanis a betárcsázáskor is hárul reá feladat. Ezt a szituációt ábrázolja a következő kép.



Betárcsázás és DHCP

Ahhoz, hogy megértsük, milyen szerepe van a Relay Agentnek, meg kell vizsgálnunk az RRAS és a DHCP viszonyát. Látni fogjuk, hogy nemcsak a továbbító ügynök tetszeleg néha a DHCP tulajdonságaival, hanem az RRAS is.
A Routing And Remote Access – ahogyan a nevében is mutatja, többek között útválasztással és betárcsázással kapcsolatos szolgáltatásokat nyújt. Amikor egy távoli, telefonos kapcsolattal rendelkező ügyfél betárcsáz, tulajdonképpen egy útválasz-

tó kel életre, amelynek egyik lába egy hálózati kártya, a másik pedig egy modem. A DHCP úgy kerül a képbe, hogy az RRAS-nak valamilyen módon gondoskodnia kell a távoli számítógép modemes lábának IP-címmel való ellátásáról.
Többféle lehetőség is létezik. Elvileg beállítható kézzel egy IP-cím a kliensnél, még egy modemes kapcsolathoz is. Na de a dinamikus címek korában? És ha egy másik hálózathoz szeretnék csatlakozni? Vagy egy nagy hálózat másik alhálózatán keresztül lépek be? A statikus cím nem igazán járható.
Egy másik megoldás lehet*(ne)*, ha a felhasználó kap egy IP-címet. Ezt megadhatjuk az ADUC felületén keresztül, de az eredmény hasonló lenne az előzőekhez. Kénytelenek leszünk az RRAS-ra bízni a cím átadását. Ekkor kezd egy router DHCP tollakkal ékeskedni.
Az RRAS kiszolgáló tulajdonságai párbeszédpanelen egy külön fül foglalkozik az IP-címek kezelésével. Íme, az alábbi képre gondolok.



Az RRAS-t és a DHCP-t ezen a panelon „lőhetjük össze"

A cím kiosztására két lehetőség is adott. A legegyszerűbb, ha mindent az alapértelmezett értéken hagyunk. Ekkor az RRAS címeket kér *(előre)* a DHCP kiszolgálótól, méghozzá 10 darabot. Egyet lefoglal magának, a többit pedig szükség szerint kiosztja a betárcsázó ügyfeleknek. Ha mind a tíz cím elfogyna, akkor kér újabb tizet és így tovább. Az alábbi címen egyébként a kikért címek száma módosítható:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\
  Services\RemoteAccess\Parameters\Ip\
  InitialAddressPoolSize
```

A címkérés az RRAS indulásakor történik. Amennyiben ekkor nem érhető el DHCP szolgáltatás, az RRAS az APIPA által meghatározott címek közül oszt ki majd az ügyfeleknek. Ha valaki 169.254.x.x címeket lát, és azt panaszolja, hogy nem tudja elérni a belső hálózatot, az gyanakodjon a DHCP és az RRAS közötti kommunikációs zavarra, és legalább egyszer indítsa újra a betárcsázó szolgáltatást úgy, hogy meggyőződött a DHCP helyes működéséről.

A fenti ábrán az is látható, hogy meghatározhatjuk azt az adaptert, ahonnan a RAS a különböző opciókat veszi. Ez kérem itt egy csapda. Az igaz, hogy az RRAS címeket kér a címkiszolgálótól. Egyéb opciókat azonban nem kér! A most emlegetett sor tehát azt mutatja meg, hogy az RRAS melyik hálózati kártyájának beállítását veszik át majd a betárcsázó ügyfelek. Az első és legfontosabb dolog tehát a RRAS LAN kártyájának nagyon korrekt beállítása. Egy rövid táblázat arról, milyen IP konfigurációs értéket honnan vesz az RRAS

Opció	Forrás
IP address	DHCP kiszolgálótól az RRAS-on keresztül
WINS server	RRAS LAN adapter beállítása
DNS server	RRAS LAN adapter beállítása
Subnet mask	Az IP-cím alapján, automatikusan Class A,B vagy C subnetet kap az ügyfél
NetBIOS scope ID	Nem továbbítódik
Node type	Ha az RRAS-nak nincs WINS szervere, akkor a kliens b-node lesz, egyébként h-node a kapcsolat ideje alatt.

No és a korábban emlegetett DHCP RRAS opciókat sem küldi át az RRAS? Nem, azokkal sem foglalkozik. Viszont az ügyfelek a kapcsolat létrejötte után küldhetnek egy DHCPINFORM csomagot, amelyben a megfelelő opciókat kérik. Itt jön a továbbító ügynök újabb feladata. Amennyiben az IP-címeket DHCP kiszolgálótól kapta az RRAS is, az ügynök automatikusan továbbítja az ügyfelek csomagjait ennek a címkiosztónak. Ha viszont statikus címlistával dolgoztunk az RRAS-on, akkor csak abban az esetben működik a továbbítás, ha az ügynököt külön konfiguráltuk, vagyis megmondtuk, hogy mely címkiosztó rendszerrel vegye fel a kapcsolatot.
A betárcsázó ügyfelek persze mit sem tudnak arról, hogy a címeket nem közvetlenül egy címkiosztó szervertől kapták. Számukra csak az a lényeges, hogy megkapják a hőn áhított IP-címet, hozzá a megfelelő paramétereket és végre kommunikálhassanak. Nekünk rendszergazdáknak azonban nem árt tisztában lennünk, hogy ez a többnyire problémamentes összekapcsolódás valójában igencsak összetett és kacifántos módon jön létre.

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

Felhasznált és ajánlott irodalom

Windows 2000 Server Internetworking Guide
Chapter 3: IP Unicast
Chapter 7: Remote Access Server
Windows 2000 TCP/IP Core Networking Guide
Windows 2000 Help – RRAS
Windows 2000 Help – DHCP

Active Directory Application Mode

Címtárat mindenkinek!

Az ADAM használatával többé nincs szükség egyedi címtárszolgáltatásra speciális, esetleg belső fejlesztésű alkalmazásaink számára sem, mivel az ADAM messzemenően képes alkalmazkodni bármilyen feladathoz. Minden ezt igénylő alkalmazás önálló, de az Active Directory-val együttműködő címtárat használhat egyedi sémával és replikációs beállításokkal.

Bevezetés

A Lightweight Directory Access Protocol *(LDAP)* alapú megoldások terjedésével, a 90-es évek közepén a vállalatok olyan üzleti szolgáltatásokat vezettek be, amelyek valamilyen címtár használatával valósították meg például a hálózati azonosítást és engedélyezést, a nyilvános kulcsok terjesztését, és az üzleti alkalmazások számos más speciális funkcióját.
Ennek eredményeképpen sok vállalatnál állt elő az a helyzet, hogy minden említett funkció támogatását külön címtár végzi, ráadásul az sem ritka, hogy ezek különböző címtár-technológián alapulnak. Például a hálózati bejelentkezéseket kezelheti a Microsoft Active Directory, a nyilvános kulcsok terjesztését X.500 címtár, az üzleti alkalmazások pedig használhatnak egy harmadik technológiát.

Miért is van több címtárunk?

Mivel minden címtár az LDAP-protokollon alapul, jogosan merül fel a kérdés, hogy miért nem használnak ezek a vállalatok egyetlen közös címtárat minden funkció számára? A válasz éppen azokban a tényezőkben rejlik, amelyek a helyzet kialakulásához vezettek:

- A címtárak közötti együttműködés hiánya – Számos címtárszolgáltatás létezik, amelyek nem képesek az együttműködésre. Például az eredeti X.500 címtárak az LDAP-protokollt sem támogatják, és még ma is számos olyan, címtárat is megvalósító termék készül, amelyek nem támogatják az LDAP-t, vagy más széles körben használt protokollt.
- A választási lehetőség hiánya – Ma is kaphatók olyan megoldások, amelyek nem képesek az összes használatos címtárszolgáltatással együttműködni, ezek vásárlói rákényszerülhetnek olyan címtárszolgáltatás bevezetésére, amelyet korábban még nem használtak.
- A koordináció hiánya – Bizonyos esetekben a vállalat különböző csoportjai egymástól függetlenül különböző megoldásokat, és ezzel együtt különböző címtár-technológiákat vesznek használatba.
- A biztonsággal kapcsolatos együttműködés hiánya – Az üzleti alkalmazások ritkán teszik lehetővé, hogy azonosítási adataikat tőlük független címtár tárolhassa, így tehát a megvásárolt új alkalmazás „hozza" a saját címtárát.

Problémák a címtárakkal

Egyre több vállalat érzi azonban a helyzet tarthatatlanságát, elsősorban a következő okok miatt:

- Nagyobb biztonsági kockázat – A címtárfüggő alkalmazások szaporodásával egyre nehezebbé válik a címtárak közötti hatékony szinkronizáció. Elengedhetetlen, hogy a vállalattal kapcsolatba kerülő, vagy kapcsolatukat megszakító munkavállalók, partnerek, megrendelők hozzáférése a vállalat hálózatához, és a különböző alkalmazásokhoz, hatékony és biztonságos módon legyen szabályozható. Ha a hozzáférés megadása lassú és körülményes, az csökkenti a termelékenységet, a hozzáférés megszüntetésének esetleges elmaradása vagy késése pedig jelentős biztonsági kockázatot jelenthet.

> Az alkalmazásokhoz tartozó egyedi címtárak szaporodása számos problémát vet fel, és jelentős többletköltséggel járhat.

- Magasabb költségek – Minden üzleti alkalmazás, amely önálló címtárszolgáltatást használ, igényli az adott technológiára kiképzett személyzetet, valamint egyedi üzemeltetési és felügyeleti folyamatok kidolgozását.
- Az üzleti folyamatokkal való integráció hiánya – A címtár-információknak az üzleti folyamatoknak megfelelően folyamatosan változniuk kell. Ha az adatokat több különböző címtárban kell párhuzamosan módosítani, automatizált eljárás nélkül szinte reménytelen feladat a konzisztencia hosszú távú fenntartása.

A vállalatoknak olyan címtármegoldásra van szükségük, amely alkalmas a hálózati infrastruktúra, és az önálló alkalmazások igényeinek együttes kezelésére. Az Active Directory kiválóan alkalmas erre a feladatra, méghozzá újabb licenckötségek és az új címtárszolgáltatás bevezetésével kapcsolatos költségek nélkül.
Az Active Directory Application Mode *(ADAM)* az Active Directory új szolgáltatása, amely a címtárat használó alkalmazások bevezetésekor számos különféle helyzetben előnyösen használható.

Az ADAM nemcsak a tartományvezérlőkön, hanem a tagkiszolgálókon, vagy önálló gépeken is futtatható, sőt több példányban is elindítható, és a példányok mindegyikéhez önálló beállításokat adhatunk meg.

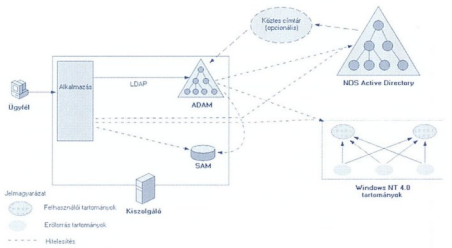
Címtár alkalmazásoknak

Sok alkalmazásnak csak egészen egyszerű címtárra van szüksége. A tárolt információkat általában csak szűk körben kell elérni, és nincs szükség a teljes szervezetre kiterjedő replikációra sem. Az alkalmazások kiszolgálása más szolgáltatási módot igényel, mint a Network Operating System *(hálózati operációs rendszer, NOS)* objektumaira vonatkozó adatok tárolása. Például, ha az alkalmazások adatainak gyakori replikációra van szüksége, a NOS-objektumokkal együtt történő tárolás jelentős terhelést jelenthet a teljes adatbázis sűrűbb replikálása miatt. Ilyen esetben az ADAM biztosíthatja az alkalmazások egyedi igényeinek megfelelő tárolást.
Az alkalmazások címtárai folyamatosan fejlődhetnek, változhat a séma és a beállítások, hogy mindig megfelelhessenek a változó üzleti igényeknek. Az ADAM példányok külön-külön módosíthatók, nincs szükség egy alkalmazás igényei miatt a teljes vállalat címtár-struktúrájának megváltoztatására.
Az ADAM kiválóan felhasználható a következő feladatok megoldására:

- Alkalmazás-specifikus címtárak
- Címtáralapú alkalmazások fejlesztése
- Az extranet hozzáférés felügyelete
- Migráció

Alkalmazás specifikus címtárak

Tekintsünk egy alkalmazást, amelynek tárolnia kell a NOS Active Directory által azonosított felhasználók bizonyos speciális adatait. Ha az adatok közvetlenül a NOS címtárba kerülnének, a teljes szervezetre kiterjedő sémamódosításra lenne szükség. Ebben az esetben a jó megoldás az, ha a felhasználók azonosítását továbbra is az Active Directory végzi, a speciális adatok tárolását pedig az ADAM-re bízzuk. Ebben az esetben az Active Directory felhasználó-azonosítási rendszere végzi az ADAM objektumokhoz való hozzáférési jogok kontrollját.



Az alkalmazás specifikus adatokat az ADAM elkülönítetten tárolja

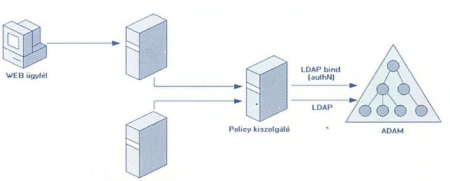
Az ADAM olyat tárolóhelyet biztosít az alkalmazás számára, ahol a „privát" adatait elkülönítetten tárolhatja, a NOS Active Directory bármiféle módosítása nélkül. Ilyen módon elkerülhető az adatok teljes körű replikációja is – az ADAM címtárhoz önálló replikációs beállítások adhatók meg. Az ADAM segítségével a címtáralapú alkalmazások, akár teljesen egyedi adatbázis sémát használhatnak adataik tárolására.
Az ADAM segítségével lehetővé válik az is, hogy a vállalat egyes szervezeti egységei az általuk kizárólagosan használt alkalmazás ADAM címtárát teljesen önállóan felügyeljék, maguk határozzák meg például a replikáció ütemezését.
Lehetőség van azonban arra is, hogy az ADAM példányok felügyeletét is a központi IT részleg végezze. A központi felügyelet alatt álló kiszolgálón lévő ADAM példányok tulajdonságai egymástól függetlenül állíthatók be. Az ADAM példányok mindegyike másik címtáralapú alkalmazáshoz tartozhat.
Az ADAM használatával a címtáralapú alkalmazások Windows NT 4.0 tartományokban is használhatók. Az ADAM ebben az esetben a Windows NT 4.0 tartomány tagjaként telepített Windows Server 2003 kiszolgálóra kerülhet, és mivel az ADAM a Windows biztonsági architektúráját használja, a felhasználók azonosítását a Windows NT 4.0 tartomány is el tudja végezni.

Címtáralapú alkalmazások fejlesztése

Az ADAM kiválóan alkalmas az Active Directory alapú alkalmazások prototípusainak üzemeltetésához, mivel az Active Directory-val megegyező programozási és felügyeleti modellt használ. Így a fejlesztők a saját számítógépükre telepített ADAM példányokkal dolgozhatnak, majd az elkészült alkalmazás egyszerűen átvihető az Active Directory-ba.
Az ADAM telepítése nagyon egyszerű, a telepítő csak minimális adatbevitelt igényel. A fejlesztők akár több különböző módon beállított és különböző adatokat tartalmazó ADAM példányt is használhatnak a fejlesztés alatt, a példányok közötti váltáshoz, még a számítógép újraindítása sem szükséges. Az ADAM-et nem kell kiszolgálóra vagy tartományvezérlőre telepíteni, hatékonyan működtethető a fejlesztők saját számítógépein, az alkalmazások fejlesztése és tesztelése a rendszergazdák közreműködése nélkül történhet. Az ADAM használatához Microsoft Windows XP, vagy a Windows Server 2003 Standard, Enterprise illetve Datacenter változatára van szükség.

Az extranet hozzáférés felügyelete

Az ADAM jól használható az extranet hozzáférés felügyeletéhez is. Az ADAM képes olyan felhasználói objektumok tárolására is, amelyeket nem a Windows hozzáférés-vezérlési rendszere azonosít. Az extranethez való hozzáférés felügyeletét ellátó Web portál alkalmazás a felhasználói adatokat az ADAM-ben tárolhatja, és ezek segítségével végezheti el a felhasználók azonosítását.

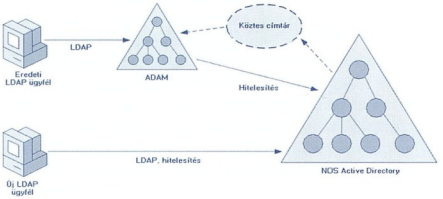


Az extranet hozzáférés felügyelete

Ez a felállás heterogén környezetben is működőképes, sőt akkor is, ha NOS Active Directory-t egyáltalán nem használunk. A webes ügyfeleket kiszolgáló portál alkalmazás tehát bármilyen platformon futhat, míg az ADAM-et egyszerűen mint LDAP elérhetőséggel rendelkező tárolót használja.

Migráció

Az ADAM felhasználható arra is, hogy a vállalat meglévő címtárát fokozatosan cserélje le az Active Directory-ra. Számos vállalat használ meglévő alkalmazásai kiszolgálásához X.500 alapú címtárat. Az Active Directory bevezetéskor az ADAM felhasználható az X.500 elnevezési struktúrát igénylő alkalmazások átmeneti kiszolgálására.
A NOS Active Directory biztosítja a szervezet megosztott biztonsági infrastruktúráját, míg az ADAM tárolja a régebbi alkalmazások adatait. Ezután a régebbi alkalmazások cseréje fokozatosan valósítható meg. Az Active Directory és az ADAM szinkronizáláshoz köztes címtárat használhatunk, amelyet például a Microsoft Metadirectory Services 2003 biztosíthat.



Migráció Active Directory-ra

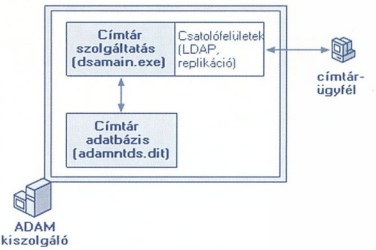
Az ADAM tulajdonságai és szolgáltatásai

Ezután sorra vesszük az ADAM használatával és felügyeletével kapcsolatos legfontosabb eszközöket és tudnivalókat:

- Az ADAM részei
- Testreszabás, bővítés
- Replikáció
- Telepítés és eltávolítás
- Több példány támogatása
- Biztonsági mentés és helyreállítás
- Felügyeleti eszközök
- Biztonság

Az ADAM részei

Az ADAM a címtárat megvalósító szolgáltatásból *(ennek kódbázisa megegyezik az Active Directory-val)*, a hozzá tartozó adatbázisfájlból, valamint a hozzáférést biztosító csatolófelületekből áll.



Az ADAM felépítése

Testreszabás, bővítés

Az ADAM támogatja a rugalmasan bővíthető adatbázis-séma használatát. A séma testreszabására a szokásos Active Directory eszközökre alapuló programok állnak rendelkezésre, például az ldifde, az ADAM Schema beépülő modul, vagy az ADAM ADSI Edit. Az egy gépen belül futó ADAM példányok is önálló sémával rendelkezhetnek.
Minden egyes ADAM példány több adatpartíciót tartalmazhat, amelyekre önálló tárolási, terjesztési és replikációs beállítások adhatók meg. A tároló biztosítja a rugalmas névhasználat lehetőségét is, használhatunk DNS vagy X.500 rendszerű elnevezési struktúrát is.

Replikáció

Az ADAM az Active Directory-val megegyező multi-master replikációs modell szerint működik, így az adatok a replikációban részt vevő bármelyik adatbázisban módosíthatók. Az ADAM hasonlóan az Active Directory-hoz lehetővé teszi a telephelyek közötti replikáció ütemezését és az átvitt adatok tömörítését is.
Ha az alkalmazások adatait az Active Directoryban tároljuk, azok replikációja csak a NOS adatokkal együtt történhet. Az ADAM használatával az alkalmazások adatainak replikációja azok speciális igényeinek megfelelően is ütemezhető.
Természetesen lehetőség van akár egy gépen belül az ADAM példányok közötti replikációra is.

Telepítés és eltávolítás

Az ADAM a Windows Server 2003 kiegészítő komponense, a Microsoft külön csomagban terjeszti azt. A csomag letölthető az **[adam]** címről. Négy letölthető fájl közül választhatunk, a szoftver létezik x86 és IA64 platformra, valamint van végfelhasználói *(retail)* és viszonteladói *(redistributable)* változat is, amelynek licence lehetővé teszi, hogy az általunk készített alkalmazásokhoz csomagolva továbbadhassuk azt.
Az ADAM a szokásos Windows alapú telepítőprogramot használja. Csak minimális adatbevitelre van szükség, és a telepítés egy script használatával könnyen automatizálható, így lehetőség van az ADAM használó alkalmazással együtt történő, teljesen beavatkozás nélküli telepítésre is.
A telepítő varázsló lehetőséget nyújt új példány, illetve meglévő példány replikájának létrehozására is.
Az eltávolító varázslóval a következő feladatokat végzi el:

- A kijelölt ADAM példányok törlése
- A beállítási adatokat tartalmazó fájlok törlése
- Partíciók törlése

Az egyszerű telepítés és eltávolítás időt takarít meg a rendszergazdák számára, és megkönnyíti az ADAM átvitelét a tesztkörnyezetből a felhasználás helyére.

Több példány támogatása

Egyetlen kiszolgálón több ADAM példány futtatható egy időben. A példányok teljesen elkülönítetten működnek, minden beállításuk egymástól függetlenül adható meg. Minden ADAM példány egyedi névvel és portszámmal azonosítható.
A több ADAM példány futtatásának lehetősége jelentős előnyökkel járhat a vállalat számára, lehetővé teszi a kiszolgálók egyesítését, könnyebbé teszi az üzleti alkalmazások fejlesztését, és az alkalmazások frissítésének végrehajtását.
Az egyes példányok a hozzájuk csatlakozó alkalmazások speciális igényeinek megfelelően állíthatók be, és az alkalmazások több különböző adattárral is együttműködhetnek.

Biztonsági mentés és helyreállítás

Az ADAM biztonsági mentése és helyreállítása a Windows operációs rendszer szokásos eszközeivel végezhető el. Minden példány könnyen automatizálható online folyamat keretében menthető, így a kritikus adatok minden körülmények között könnyen és gyorsan hozzáférhetők lesznek. Az NTBackup segédprogram lehetővé teszi az ADAM példányok online mentését és offline helyreállítását.

Felügyeleti eszközök

Mivel az ADAM az Active Directory egy üzemmódjának tekinthető, a felügyelet az Active Directory-hoz hasonlóan, az ismert felügyeleti eszközök módosított változataival végezhető el. Az ADAM felügyeleti eszközei az alkalmazással együtt kerülnek telepítésre:

- **Ldp** *(Ldp.exe)* – az ldp segítségével *(ami grafikus felhasználói felülettel működik)* LDAP műveleteket végezhetünk az ADAM címtáron.
- Az **ADAM ADSI Edit** az ismert ADSI Edit eszközön alapul, és lehetővé teszi a címtár valamennyi objektumának *(a sémának és a beállítási adatoknak is)* megjelenítését, az objektumok módosítását és a hozzáférés vezérlési listák szerkesztését.
- A szokásos monitorozó eszközök *(például a **PerfMon**)* használhatók az ADAM-mel kapcsolatos hálózati és rendszerteljesítmény nyomon követésére. Minden egyes ADAM példány teljesítményadatait naplófájlokba gyűjthetjük, és a Microsoft Management Console *(MMC)* változatos megjelenítési képességeit felhasználva elemezhetjük azokat.
- A **Dsadmin** és a **dsdbutil** *(az ntdsutil-hoz hasonló)* használható az adatbázis karbantartásához, az egyedi főkiszolgáló-műveletek *(single master operation)* végrehajtásához, és a címtár-partíciók létrehozásához.

Jelentős megtakarítással járhat, hogy az Active Directory-hoz nagyon hasonló ADAM felügyeleti eszközök használatához nincsen szükség külön képzésre.

Biztonság

Az ADAM szorosan illeszkedik a Windows operációs rendszer biztonsági modelljébe. Az ADAM-ben tárolt objektumokhoz és a szolgáltatáshoz magához a következők férhetnek hozzá:

- A NOS Active Directory-ban szereplő felhasználók
- A Windows NT 4.0 tartományban szereplő felhasználók
- A helyi számítógépen felvett felhasználói fiókok

Az ADAM támogatja továbbá az LDAP alapú hitelesítést külső biztonsági infrastruktúra használatával. Létrehozhatunk felhasználói fiókokat az ADAM-en belül is, így az alkalmazások a címtárra bízzák a hitelesítést, míg a jogok kiosztását maguk végzik el. Ebben az esetben az ADAM a hitelesítést egyszerűen az LDAP csatolás segítségével végzi el.
A címtárobjektumokhoz való hozzáférés szabályozása a Windows meglévő hozzáférés vezérlési listáinak *(Access Control List, ACL)* felhasználásával történik. Ilyen módon lehetőség van a hozzáférés részletes szabályozására valamennyi ADAM példány minden objektumára vonatkozóan. Az alkalmazások tovább bővíthetik a hozzáférés szabályozás lehetőségeit saját keretrendszer felhasználásával, míg a felhasználók hitelesítését továbbra is a címtár végzi.

Összefoglalás

Azok a vállalatok és fejlesztők, akiknek címtárszolgáltatásra van szükségük alkalmazásaik számára, az Active Directory új, kiegészítő szolgáltatása révén számos előnyhöz juthatnak:

- Egyszerű telepítés – A fejlesztők és végfelhasználók könnyen telepíthetik LDAP szolgáltatásként a Windows 2003 Server különféle változataira és a Windows XP-re. Az ADAM telepítése, újratelepítése és eltávolítása könnyen automatizálható, így kiválóan megfelel az alkalmazással együtt terjesztett címtárszolgáltatás céljára.
- Alacsonyabb infrastrukturális költségek – Ha egységes címtár-technológiát használunk az alkalmazásokhoz és a hálózat objektumaihoz, csökkenthetőek az infrastruktúra fenntartásával és a felügyeletet végző személyzet képzésével kapcsolatos költségek.
- Magasabb biztonsági szint – A Windows biztonsági modelljével való integráció lehetővé teszi minden ADAM-et használó alkalmazás számára, hogy a felhasználók hitelesítését a vállalati Active Directory-ra bízza.
- Nagyobb rugalmasság – A fejlesztők könnyen készíthetnek címtáralapú alkalmazásokat anélkül, hogy a vállalat címtár-sémáját módosítani kellene.
- Megbízhatóság és skálázhatóság – Az ADAM-et használó alkalmazások az Active Directory-nál megismert megbízhatóságot, skálázhatóságot és teljesítményt kapják az adataikat tároló címtártól.

Az ADAM segítségével egyetlen címtár-technológiát használhatunk a vállalatnál előforduló minden címtárat igénylő feladat megoldására.

Szerényi László
szerenyi.l@met.hu

Tanúsítványkiadók a Windowsban

Kulcsarchiválás és –helyreállítás, adatmentés, Certificate Trust List

A Windows Server 2003, Enterprise Edition tanúsítványkiadó szolgáltatása segítségével biztonságosan tárolhatók és szükség esetén vissza is tölthetők a felhasználók privát kulcsai. Cikkünkben kitérünk még a tanúsítványkiadó adatbázisának biztonsági mentésére, valamint a tanúsítványláncok összekapcsolásának egyik módjára.

Miért van szükség a privát kulcsok biztonsági jellegű tárolására?

Nem téved, aki úgy tudja, hogy a nyílt kulcsú biztonsági architektúrában a felhasználó privát kulcsa csakis a felhasználóé, ahhoz senki más, semmilyen formában nem férhet hozzá. A privát kulcs titkosítva a felhasználó profiljában, biztonságosabb megoldásoknál hardveres kulcstároló eszközön található, utóbbit a kulcs még „erőszakkal" sem hagyja el. Ott keletkezik, és ott is éli le teljes életét.

A felhasználó privát kulcsának megsemmisülése azonban komoly következményekkel járhat. Ha a kulcshoz tartozó tanúsítványt digitális aláíráshoz használtuk, még nem túl nagy a gond *(egy új kulcspárral és tanúsítvánnyal továbbra is írhatunk alá digitálisan, bár az új kulcs publikus felét, azaz az új tanúsítványt közzé kell tenni és eljuttatni a digitális aláírást ellenőrző partnerekhez)*. Ha viszont a megsemmisült kulcspárt titkosításhoz használtuk, nagy a baj: minden adatot a megsemmisült privát kulcs publikus párjával titkosítottunk, így azt csak a privát kulcs segítségével lehet*(ne)* visszaállítani. A titkosító kulcspár privát felének elvesztése tehát automatikusan a titkosított adatok elvesztését is jelenti! *(Éppen ezért nem szokták a titkosító kulcspárt olyan hardvereszközön tárolni, ahonnan katasztrófa esetén sem lehet kiexportálni. Különben, ha elveszik az eszköz – átmegy rajta a villamos –, elvesznek az adataink.)*

Központi kulcstárolás

A megoldás az, hogy – ha a privát kulcs exportálható –, a felhasználó saját munkamenetéből, saját magának, jelszóval védett és titkosított .pfx fájlba exportálhatja a tanúsítványt és a hozzá tartozó kulcspárt. Így az később a jelszó ismeretében bármikor visszaállítható.

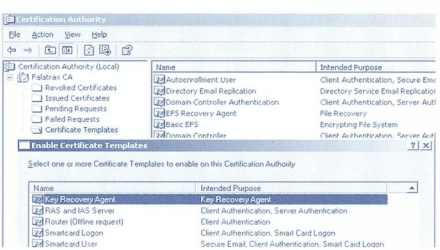
Ahogy azonban a PKI megoldások világszerte egyre inkább terjednek, úgy lesz egyre nagyobb azon felhasználók száma, akinek a tanúsítvány exportálása már túl nagy, már-már megoldhatatlan feladatot jelent. Az „átlagos" felhasználó képes arra, hogy bekattintsa a levelezőprogramjában a levél digitális aláírására vagy titkosítására utaló opciót, de általában – és sajnos – ismeretlen számára a tanúsítványok, kulcsok, tanúsítványkiadók világa. Nem véletlen, hogy a Windows Server 2003 Enterprise Edition tanúsítványkiadói már támogatják az automatikus tanúsítványkiadást *(az előző részekben már ejtettünk szót erről)*. A következő lépés, bármilyen furcsán hangzik is, a felhasználók privát kulcsainak központosított, biztonságos tárolása.

Ebben az esetben a felhasználók privát kulcsai a tanúsítványkiadó adatbázisába is bekerülnek. A biztonsági szempontból legfontosabb kérdés, hogy milyen formában történik ez az adattárolás, mi az, ami megakadályozza, hogy a felhasználók privát kulcsai illetéktelen kezekbe kerüljenek?

A kulcshelyreállító ügynök

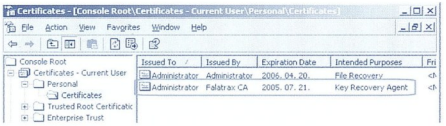
A bekezdéscím elárulja: a biztonságot nem „mi", hanem „ki" garantálja. Amikor a felhasználó olyan tanúsítványt kér, amelynek sablonjában bekapcsoltuk a privát kulcsok archiválását, a tanúsítványkiadó szervezet a felhasználó *(számítógépének)* elküldi a kulcshelyreállító ügynök*(ök) (Key Recovery Agent)* kifejezetten erre a célra készült, kulcshelyreállító tanúsítványának publikus kulcsát. A felhasználó az ügynök publikus kulcsának segítségével titkosítja a saját privát kulcsát, majd elküldi azt a tanúsítványkiadónak, ami a titkosított adatot tárolja az adatbázisban. Az adatbázisban tárolt felhasználói privát kulcs csakis a kulcshelyreállító ügynök tanúsítványának privát kulcsa segítségével nyerhető ki. A kulcshelyreállító ügynök tehát bizalmi állás *(akárcsak az EFS titkosított fájlok helyreállítására képes ügynöke, a File Recovery Agent)*.

A kulcshelyreállító ügynök tanúsítványát egy speciális tanúsítványsablonból *(Key Recovery Agent)* készítjük. A Windows tanúsítványkiadó telepítés után automatikusan ilyen sablonból tanúsítványt nem ad ki, ezért „kézzel" fel kell vennünk azt a kiadható tanúsítványsablonok közé:



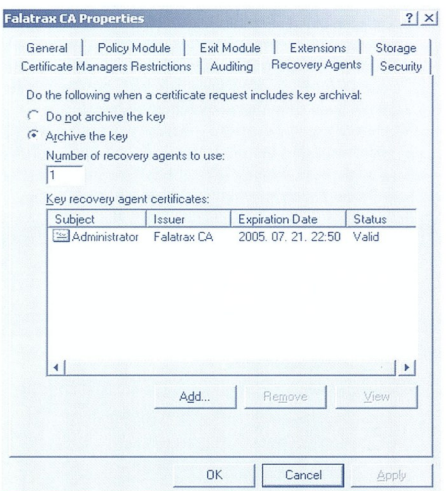
A Key Recovery Agent tanúsítványsablon felvétele

Miután ezzel megvagyunk, következhet a tanúsítványkérés. A kulcshelyreállító ügynök*(ök)* személye cégenként más és más lesz, de azt mindenképpen vegyük figyelembe, hogy ez a tudás tényleg hatalom, hiszen segítségével bármely felhasználó privát kulcsához hozzá lehet férni. Miután kijelöltük az ügynököket, a megszokott módon kérjünk részükre Key Recovery Agent tanúsítványt!



Key Recovery Agent tanúsítvány az ügynök profiljában

A következő lépésben a tanúsítványkiadó szolgáltatásba kell feltöltenünk a kulcshelyreállító tanúsítvány publikus kulcsát. A tanúsítványkiadó ezeket a tanúsítványokat küldi majd el a felhasználónak, aki majd a bennük található publikus kulccsal titkosítja a saját privát kulcsát és küldi el a tanúsítványkiadó adatbázisába. A Key Recovery Agent ügynökök telepítése a Certification Authority eszközben, a tanúsítványkiadó tulajdonságlapjának Recovery Agents oldalán végezhető el:



Kulcsarchiválás és Recovery Agent tanúsítványok beállítása a tanúsítványkiadó szolgáltatásban

Ne ijedjünk meg, ha a felvett tanúsítványok státusza először „Not loaded". Az Apply gombra kattintva a konzol újraindítja a tanúsítványkiadó szolgáltatást, és legközelebb már a „Valid" státusszal fogunk találkozni.
Ha megnézzük a tanúsítványkiadó szolgáltatást futtató számítógép tanúsítványtárát, ott a KRA mappában láthatjuk is a betöltött tanúsítványokat. Ha egy ilyen tanúsítványt megnyitunk, az is látszik, hogy a gép nem rendelkezik a tanúsítványhoz tartozó privát kulccsal.
Miután a tanúsítványt betöltöttük, még egy dolgunk van: fontos, hogy ha a kulcshelyreállító tanúsítványt nem valamilyen hardvereszközön tároljuk, a kulcshelyreállító ügynök profiljából privát kulccsal együtt exportáljuk, majd töröljük a kulcshelyreállító tanúsítványt. A kiexportált .pfx fájlt tároljuk biztonságos helyen *(például egy páncélszekrényben)*, hiszen arra csak akkor lesz szükség, ha egy archivált privát kulcsot helyre kell állítanunk.
Ezzel a tanúsítványkiadó kész a kulcsok archiválására.

Kulcsarchiválás bekapcsolása a tanúsítványsablonban

A tanúsítványkiadó csak olyan tanúsítványok esetén menti a felhasználó privát kulcsát, amelyek sablonjában ezt a szolgáltatást kifejezetten bekapcsoltuk. Miután itt a tanúsítványsablonok szerkesztéséről van szó, rögtön látható, hogy a kulcsarchiválás csakis az új, v2-es tanúsítványsablonokkal, Windows Server 2003 Enterprise Edition kiszolgálóra telepített tanúsítványkiadó szolgáltatás esetén működik.
Válasszunk tehát egy tanúsítványsablont *(példánkban a korábban automatikus kiadásra már előkészített Autoenrollment User sablont használjuk erre)*, és nyissuk meg a sablon tulajdonságlapjának Request Handling oldalát!

A kulcsarchiválás bekapcsolása a tanúsítványsablonban

Ezzel készen is vagyunk. A sablonból készült tanúsítványok kérésekor a tanúsítványkiadó automatikusan archiválja a privát kulcsokat.

Kulcshelyreállítás

Tegyük fel, hogy bekövetkezik az első katasztrófa: a felhasználó elveszíti a privát kulcsot *(törlődik a profilja, vagy mondjuk a Smart Cardja beleesik a WC-be..)*. A helyreállításhoz szükségünk lesz a kulcshelyreállító ügynökökre *(pontosabban a tanúsítványaikra)*. Jelentkezzünk be a tanúsítványkiadó kiszolgálóra, és töltsük be az összes szükséges kulcshelyreállító tanúsítványt. A privát kulcs helyreállításához két eszköz áll rendelkezésünkre, ezek közül az első a minden CA-n megtalálható certutil parancs.

A helyreállítás két lépésből áll:

- Először meg kell keresnünk a tanúsítványkiadó adatbázisában a megfelelő titkosított adatcsomagot *(blob)*, majd elmenteni azt
- Ezután a kulcshelyreállító tanúsítvány segítségéből az elmentett blob-ból kinyerni a kulcsot és .pfx fájlba menteni azt.

Az első lépésre azért van szükség, mert az adatbázisban adott esetben többezer elmentett privát kulcs is lehet. Ezek közül kell kiválasztanunk azt az egyet, amit keresünk. Ehhez a certutil –getkey parancsot használjuk:

```
C:\>certutil -getkey falatrax2003\teszt
↳ tesztcert.bin
```

A –getkey opció utáni első paraméter a keresett tanúsítvány azonosítója, ez lehet:

- a tanúsítványban tárolt Common Name
- a tanúsítvány sorozatszáma *(Serial Number)*
- a tanúsítvány azonosító hash-e *(Thumbprint)*
- a tanúsítványt kérő felhasználó neve *(domain\usernév formában)*
- a tanúsítványt kérő felhasználó UPN neve *(usernév@domain formában)*

A második paraméter egy fájlnév, ahova sikeres keresés esetén a bináris adatok kerülnek majd. *(Részletesebb információkat a certutil -getkey -? parancs segítségével kaphatunk).* Ha a bináris fájl elkészült, következik a dekódolás a certutil -recoverkey parancs segítségével:

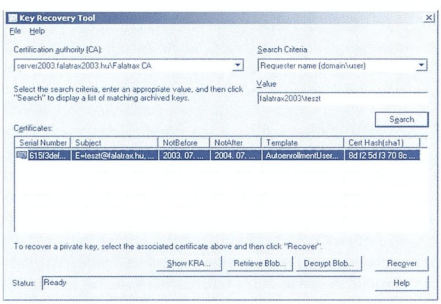
```
C:\>certutil -recoverkey -p titok tesztcert.bin
 tesztcert.pfx
```

A -p kapcsoló után megadhatjuk a leendő .pfx fájl titkosításhoz használt jelszót *(itt: „titok")*, majd a bináris adatot tartalmazó fájl neve, azután pedig a létrehozandó .pfx fájl neve következik. Ha a kulcshelyreállító tanúsítvány*(ok)* elérhetők, a parancs dekódolja a privát kulcsot és létrehozza a .pfx fájlt, amit már csak el kell küldenünk a felhasználónak, aki importálhatja azt.

Key Recovery Tool

Ha letöltjük a Windows Server 2003 Recovery Kit Tools [w2k3rktools] csomagot, egy grafikus eszköz könnyíti meg a kulcshelyreállítást: ez a Key Recovery Tool *(krt.exe).*

A Key Recovery Tool működés közben



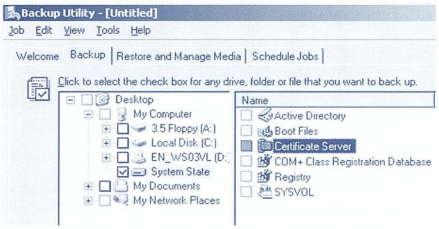
A Key Recovery Tool is a certutil parancsot használja a háttérben, a működési elve tehát ugyanaz. Először kiválasztjuk a tanúsítványkiadót *(Certification authority)*, ezután megadjuk a keresési feltételeket *(Search Criteria, Value).* A Search gombra kattintva elkezdődik a keresés, és a Certificates listában láthatjuk a találatokat. Ezután már csak ki kell jelölnünk a helyreállítani kívánt tanúsítványt, és a Recover gombra kattintani. Adjuk meg a .pfx fájl nevét *(ez alapértelmezésben a tanúsítvány sorozatszámából készül – okos ötlet),* majd a .pfx fájl titkosításához szükséges jelszót, és már készen is vagyunk!

A tanúsítványkiadó biztonsági mentése

Katasztrófa azonban nemcsak a felhasználónál, de a kiszolgálón is bekövetkezhet. A tanúsítványkiadó szolgáltatás helyreállításához két nagyon fontos dologra van szükségünk:

- A tanúsítványkiadó szolgáltatás tanúsítványára
- A tanúsítványkiadó adatbázisára

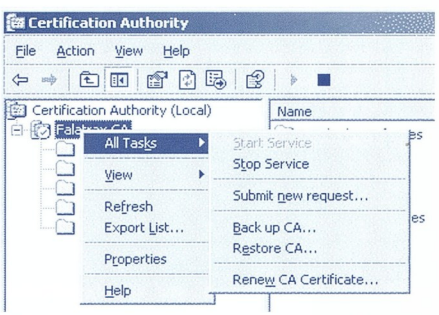
Ezek mindegyike szerepel a Windows teljes rendszermentésében *(System State)*:



A teljes rendszermentés tartalmazza a tanúsítványkiadó szolgáltatást is

Rendszerállapotot azonban csak egy-az-egyben lehet visszatölteni, azaz a Certificate Server esetleges sérülése esetén a System State mentés a CA adatbázisa mellett visszatöltené az Active Directory, a Registry, és egyebek régebbi állapotát is. Erre nem biztos, hogy szükségünk van, ezért a System State mentés mellett „kézzel" is mentsük el a tanúsítványkiadó adatait:

- Exportáljuk *(privát kulcsostul)* .pfx fájlba a tanúsítványkiadó tanúsítványát *(a számítógép tanúsítványtárában megtaláljuk).* A .pfx fájlt tartsuk zárt, biztonságos helyen, hogy ahhoz senki ne férhessen hozzá!
- Rendszeresen készítsünk mentést a tanúsítványkiadó adatbázisáról a tanúsítványkiadó konzol Back up CA... parancsával! *(Ez is tartalmazza a privát kulcsot, a tanúsítványt és természetesen az adatbázist is)*



A tanúsítványkiadó adatbázisának mentése

A tanúsítványkiadó szolgáltatás helyreállítása

A tanúsítványkiadó adatbázisa a Restore CA... paranccsal korábbi mentésből bármikor újratölthető.

Ha a tanúsítványkiadó szolgáltatást újra kell telepítenünk, a telepítés során a Public and Private Key Pair oldalon válasszuk ki a Use an existing key opciót, majd az Import... gombra kattintva importáljuk be a korábban elmentett kulcsot a .pfx fájlból. A telepítés befejezését követően a tanúsítványkiadó – üres

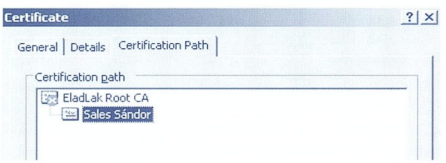
adatbázissal – elindul, ezután következhet a Restore CA... parancs, és az adatbázis visszatöltése.

Tanúsítványláncok összekapcsolása

Tegyük fel, hogy cégünk, az ingatlanépítéssel foglalkozó Falatrax Kft. két évre szóló stratégiai partnerséget köt az EladLak Virtuális Ingatlanforgalmi Kft-vel *(VIKft)*. A tervek szerint a biztonságos levelezés érdekében a két cég között digitális aláírással, illetve titkosítással felvértezett elektronikus levelezés zajlik majd. A tanúsítványok kezelése azonban kérdéses: hogyan vesszük rá például a Falatrax nyílt kulcsú biztonsági architektúráját arra, hogy elfogadja és érvényesnek tekintse az EladLak saját CA-i által kiadott tanúsítványokat? Ha a Falatrax Trusted Root Certification Authorities listájába felvesszük az EladLak gyökér CA-ját, a dolog működni fog. Csakhogy: ebben az esetben minden EladLak CA által, ráadásul bármilyen célra kiadott tanúsítvány automatikusan érvényes lesz a Falatrax-nál, ezt pedig nem szeretnénk. Ha korlátoznunk kell egy partner CA által kiadott tanúsítványok érvényességi körét és -idejét, a megbízott gyökérkiszolgálóként való felvétel helyett – Windows 2000-en futó CA-k esetén is – használhatjuk a Certificate Trust List *(CTL)* funkciót.

A Certificate Trust List

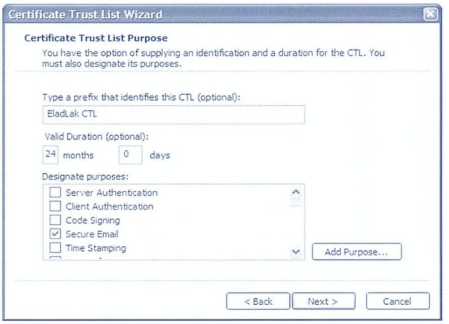
A CTL egy digitálisan aláírt lista, amely külső, adott célokra és adott időszakban megbízhatónak minősített tanúsítványkiadók tanúsítványait tartalmazza. *(A CTL-t egy általunk megbízott – célszerűen a saját – tanúsítványláncból származó speciális, „Trust List Signing" application policyt tartalmazó – például Trust List Signing sablonból készült - tanúsítvánnyal kell aláírni, ez bizonyítja majd a lista érvényességét).* Amikor a tanúsítvány ellenőrzése során eljutunk egy gyökér CA-ig, de az nincs benne a megbízott gyökérkiadók listájában, a Windows ellenőrzi, hogy nincs-e olyan CTL, ami tartalmazza a kérdéses CA-t. Ha van, és a CTL digitális aláírásához használt tanúsítvány érvényes, a kérdéses tanúsítvány is érvényes: de csak az érvényesítéséhez felhasznált CTL által meghatározott célokra és időszakban.



Itt a tanúsítvány még érvénytelen: az EladLak Root CA nem szerepel a megbízható gyökértanúsítványkiadók között

CTL-t vállalati szinten *(a csoportos házirend segítségével)*, de akár egy különálló számítógépen *(Windows XP-n is)* definiálhatunk. A vállalati szintű definícióhoz a Group Policy Objektumban a Computer illetve User csoport Windows Settings→ Security Settings → Public Key Policies → Enterprise Trust elemhez navigáljunk, majd válasszuk a New... parancsot. Ha a saját, tartományon kívüli számítógépünkön szeretnénk valamiért CTL-t létrehozni, indítsunk egy MMC konzolt, majd töltsük be a Group Policy Object Editor modult. Ekkor a User csomópont alatt *(itt ugyanis csak felhasználónak lehet CTL-je)*, a Windows Settings → Security Settings → Public Key Policies → Enterprise Trust elemnél hozhatunk létre új CTL-t, illetve importálhatunk meglévőt.

Új CTL létrehozásakor Certificate Trust List Wizard első oldalán elnevezhetjük a CTL-ünket *(ennek csak kozmetikai jelentősége van)*, illetve megadhatjuk a CTL érvényességét, és az engedélyezett tanúsítvány-szerepköröket:



A CTL varázsló és a legfontosabb beállítások: érvényességi idő és tanúsítvány-szerepkörök

Ezután ki kell választanunk a megbízott Root CA-kat, majd a CTL digitális aláírásához használt tanúsítványt. Végül ha akarjuk, időbélyeget is elhelyezhetünk a CTL-ben.

A példabeli EladLak CTL érvényre jutása után a Falatrax-nál az EladLak Root CA által kiadott, addig érvénytelen tanúsítványok érvényesek lesznek. A tanúsítvány tulajdonságlapján pedig megnézhetjük az érvényességi láncot, amely tartalmazni fogja a CTL-t és az azt aláíró tanúsítványt is:

CTL in Action: a tanúsítvány immár érvényes. (A CTL aláírója az Administrator volt)

A CTL tündöklése és halála

A Windows Server 2003 azonban ezen a téren is újat hozott. A CTL használata – bár működik – már nem ajánlott, ehelyett egy új fogalmat kell megismernünk, ami nagyjából ugyanezt a célt szolgálja, mégis sokkal jobb, sokkal összetettebb: ez a Qualified Subordination *(azaz kb. „minősített altanúsítványkiadók")*, a következő rész témája.

Fülöp Miklós
mick@inetcom.hu

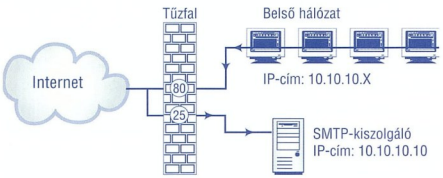
A cikkben szereplő URL-ek a http://technet.netacademia.net/go?kulcsszó címen érhetők el.

Háromlábú ISA Server

DMZ külön hálózaton

Amikor egy cég „magához ragadja" az Internetes szolgáltatások körét, és saját SMTP, HTTP és egyéb kiszolgálókat üzemeltet, mindig felmerül az a nem is egyszerű tervezési kérdés, hogy hova tegyük a kiszolgálókat? A belső hálózatra, és onnan publikáljuk? Ez a jelenlegi, férges korszakban veszélyes lehet. Két tűzfal közötti DMZ-be? De hisz az plusz egy vasat igényel! Vagy netán használjuk a háromlábú modellt? Miért ne?

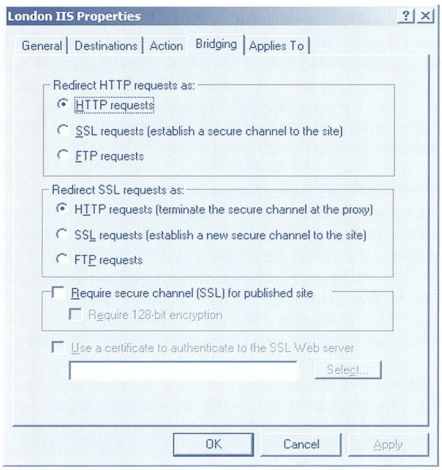
A bevezetőben felvetett tervezési dilemmát – mint láttuk - több szempont szerint oldhatjuk fel. A kérdés csak az: melyik ujjamat harapjam? Ha az első megoldás mellett döntünk, és kiszolgálóikat egyszerűen kipublikáljuk a belső hálózatról, megsértjük azt a szabályt, hogy kéretlen külső forgalmat egyáltalán ne engedjünk be. Pedig a leggyakrabban ezt szokták választani, mert ehhez elég egy tűzfal, két hálózati kártya és egyetlenegy publikus IP-cím. Az alábbi ábra ezt az esetet szemlélteti:



(SMTP) kiszolgáló publikálása a belső hálózatról egy tűzfal esetén

Csakhogy ezzel a lépéssel a publikált szerver felé ugyan, de lyukat ütöttünk a falon, amin át idegenek nyomakodhatnak a belső hálózatunk felé. Nem véletlen, hogy az ISA Serverben pontosan az SMTP, a POP3 és a DNS protokollokra vannak betörésdetektáló szűrők: ezeket a portokat szoktuk kinyitni. *(Nomeg a 3389-es, Terminal Services portot. RDP-szűrő nincs az ISA Serverben, más kérdés, hogy ezidáig még nem is lett volna rá szükség, mert [egyelőre] nincs ismert RDP-támadás.)* Természetesen a HTTP-forgalom is átesik egyfajta szűrésen, mert az meg átmegy a Web Proxy szolgáltatáson, ami azt jelenti, hogy nincs közvetlen kapcsolat a két valódi kommunikáló fél között, minden csomagot átvizsgál és átalakít a proxy. Erre ékes bizonyíték lehet az SSL-láncolás, ugyanis az ISA rávehető arra, hogy a beérkező sima HTTP-ből a belső kiszolgáló felé HTTPS-t *(SSL)* csináljon, de ami meglepőbb lehet, ennek a fordítottjára is képes. Kívülről jövő HTTPS-ből sima HTTP-t csinál, ha úgy akarjuk.

Erre nem azért képes, mert „meghekkeli" a rajta átmenő SSL-csatornát, hanem mert nem is megy át rajta a HTTPS – a csatorna nála végződik.



Hogyan továbbítsuk az SSL-t? Simán? És hogyan a pucér HTTP-t? SSL-csatornában?

LAT-olgassunk

Honnan tudja az ISA, hogy melyik lába „lóg kifelé"? Ezt a tűzfal telepítésekor mondjuk meg neki a Local Address Table *(LAT)* kitöltésével. A LAT-ba felvett címek jelentik a tűzfal számára a belső címeket. Ami nincs bent a LAT-ban, az definíció szerint külső cím, tehát hallgass a neve. Tulajdonképpen az egész tűzfal mély hallgatásba burkolózik a külső kérések elől – amíg valamit ki nem publikálunk.

Most tételezzük fel a legrosszabbat: a publikált kiszolgálónk kívülről elérhető szolgáltatására valaki „kifejleszt" egy ügyes buffer overrun támadást, másvalaki pedig férget is fabrikál hozzá. Ne nevessünk korán, a januári Slammer pont ezt a forgatókönyvet követte: buffer overrunnal megdöntött egy kívülről elérhető szolgáltatást, az SQL Server portátirányítóját. És már kész is a baj. Ha bejut a féreg, mindjárt a belső hálózatunkon találja magát, és lesz nemulass!

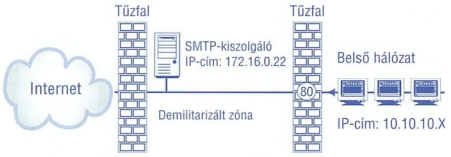
GIF-féreg, JPEG-vírus

Hadd tegyek itt egy kis szakmai kitérőt azok számára, akik szerint mindez lehetetlen, mert a támadást vagy megfogja az ISA beépített filtere *(már amennyiben engedélyezve van)*, vagy

mire átvergődik a NAT-on, minden erejét elveszíti, tehát hatástalan lesz.
Tisztelettel jelentem, hogy feltaláltam a GIF-féregvírust. Ez GIF-képállományokban rejtőzik, és oly módon hat, hogy buffer overrunt okoz a GIF-nézegető alkalmazásban, mondjuk az Internet Explorer megfelelő moduljában. Az ötletet ingyen adom, már csak egy fejlesztő hiányzik, aki megírja.
De mi az a buffer overrun, ami annyi gondot okoz napjainkban? Ha egy alkalmazás bemeneti adatot vár a külvilágból, azt gyakran lokális változóba, tehát a verembe *(stack)* teszi. Ha nem figyelünk oda, a beérkezett adatok olyan területeket is felülírhatnak, ahol már egy függvényhívás visszatérési címei vannak a veremben *(EIP)*, így az adatot váró függvényünk nem tud visszatérni a hívójához. Megfelelő kreativitással úgy vágjuk felül a vermet, hogy a visszatérési cím ismét a veremre mutasson, ahová ismét csak saját kódokat helyezünk el, ami azután gyönyörűen lefut.
Pusztán azért tudhatjuk viszonylagos biztonságban magunkat, mert a vírus- és féreggyártók általában minimális ismeretekkel *(sem)* rendelkeznek a megtámadott rendszerekről, és ezt a hiányosságukat nem is óhajtják bepótolni. *(Lásd a Nimdát, aki nem volt képes megfertőzni nem C:\-ra telepített operendszereket.)*
Egy szó mint száz, a veszély valós: belső hálózatról ne publikáljunk semmit. Két másik tűzfalszervezési modell terjedt el a gyakorlatban, melyek mindegyike figyelembe veszi ezt a veszélyt, és demilitarizált zónába *(DMZ, vagy más néven screened network, harmadik nevén perimeter network)* helyezi a külvilág felé felkínált kiszolgálókat. Hogyan? Az egyik modell egy további tűzfalat, a másik pedig egy további alhálózatot *(és IP-címeket)* igényel.

Back-to-back DMZ

Ebben a felállásban két tűz*(fal)* közé „szorulnak" a publikálandó kiszolgálók az alábbi ábrának megfelelően:

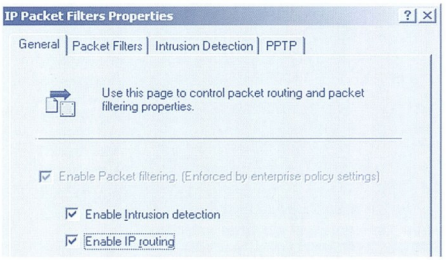


DMZ a két tűzfal között

Tulajdonképpen nem jogos, sőt félrevezető a back-to-back elnevezés, mert a két tűzfal nem egymásnak háttal áll, hanem éppen hogy egyirányba „néznek". Mindegyikük a fenti ábrán tőle balra eső hálózatot tekinti külsőnek, a jobbra esőt pedig belsőnek. Tehát a DMZ a baloldali számára **belső** hálózat *(benne van a LAT-ban)*, a jobboldali számára viszont **külső** *(nincs benne a LAT-ban)*.
Amire még felhívnám a figyelmet, az a DMZ-ben használt IP-cím. Ez bizony belső cím, akárki akármit mond is. Az SMTP-kiszolgáló úgy jut ki a netre, hogy a baloldali tűzfalon hagyományos módon publikáljuk őkelmét.
Tehát ebben a modellben a következő történt: mivel a publikált hálózat veszélyeket rejt magában, a munkaállomásokat egy újabb tűzfallal leválasztottuk erről a veszélyes alhálózatról. A második tűzfalon viszont már nem publikálunk semmit. Praktikus haszna nincs ugyan, de ilyenformán akár tucatnyi tűzfalat is egymás mögé állíthatunk *(kisvonat modell)*, köztük seregnyi DMZ-vel.
A kialakítás egyértelmű hátránya a sok-sok plusz kábelen felül az eggyel több tűzfal, bár ebből a hátrányból sokan úgy kovácsolnak előnyt, hogy a két tűzfal különböző gyártmányú, így ha a külső el is esik, a belső még mindig állja az ütéseket.
Vajon hogyan lehetne a két tűzfalas megoldáshoz hasonló biztonságot elérni egyetlen tűzfallal? Úgy, hogy a publikálandó kiszolgálókat mégsem rakjuk be a LAT-ba. Hanem hova?

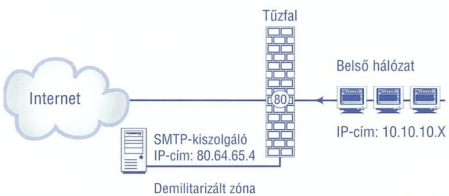
A háromlábú ISA

Ki mondta, hogy egy tűzfalnak csak két hálózati kártyája lehet? Hogy csak egy publikus hálózattal állhat kapcsolatban? Ez nem így van. Lehet több külső hálózattal is kapcsolata, ekkor ezek között útválasztási feladatot lát el *(router)*. Ehhez egyetlenegy pipát kell bepöccinteni az ISA konzolban az IP Packet Filters ágon:



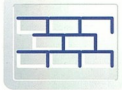
Az ISA csomagtovábbító funkciójának bekapcsolása

Tulajdonképpen nem egy valódi routert kapunk, hanem egy bezártat. Csak azokat a protokollokat továbbítja, és csak oda, amiket a Packet Filterekben megadunk. Tehát kezdetben a következő csomagokat engedi át: semmit. Ez igazán bíztató!
Most pedig oszlassunk el egy közkeletű tévedést: háromlábú ISA esetén a DMZ **nem** a belső hálózatra esik *(nincs benne a LAT-ban)*! Sőt! Egyenesen kívül kell lennie, mert ha belül lenne, éppúgy fenyegetné jelenlétével a belső számítógépeket, mint a legelső, legegyszerűbb modellben. Az ISA Server ugyanis a LAT-ban szereplő *(tehát belső)* hálózatok között korlátlan routolást végez. Korlát csak kívül van! Tehát a modell így fest:



Háromlábú ISA Server esetén a DMZ publikus IP-címet kap, tehát kívül van!

Ami a legfontosabb különbséget illeti *(s ez egyben a modell legeslegnagyobb hátránya)*: ebben a DMZ-ben minden kiszolgálónak publikus IP-címmel kell rendelkeznie, különben nem „útválasztódnak" hozzá a csomagok. Itt nincs NAT, sőt sem-



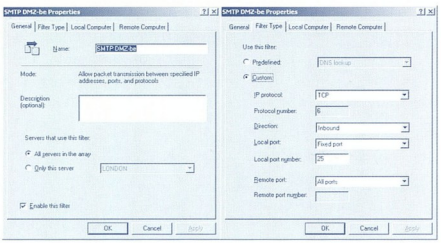
milyen címfordítás, csupán *(szűrt)* csomagtovábbítás. Nincs publikálás és PubVarázsló, csak packet filterek engedélyezése-tiltása.

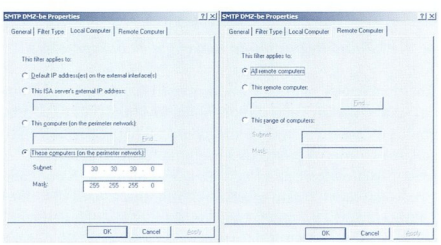
„Publikáljunk" SMTP-t!

Azért tettem idézőjelbe a „publikáljunk" szót, hogy ezzel is hangsúlyozzam: ez nem a megszokott Server Publishing lesz, csupán át kell engedni a packet filteren az SMTP-forgalmat. Jobban hasonlít router konfigurálására, mint a hagyományos publikálásra.

Az SMTP-levelek betalálásáról az MX-rekordok gondoskodnak, de ellentétben a Server Publishinggel, itt az MX nem az ISA-ra, hanem közvetlenül az SMTP-kiszolgálóra mutat. Az ISA „csak" egy router – egyelőre egy bezárt router.

Most tehát egy olyan filtert kell alkotni, amely a világ bármely pontjáról **beérkező** 25-ös porti forgalmat elengedi a végállomásig. Idemásolom a megfelelő filter négy lapját:





És készen is vagyunk. A levelek szépen betalálnak. Csak éppen válaszolni nem tudunk, ki már nem mennek. Vajon miért? Mert a fenti filter nem engedélyezi az ellentétes irányú SMTP forgalmat, még akkor sem, ha a második fülön „Both" irányt választunk! Hogyhogy? Inbound + Outbound <> Both?

De nem ám! Ugyanis a packet filterek beállításának mind a négy füle összefügg, és értelmezése a következő:

- ha a levél Remote gépről Local gépre tart *(DMZ-be bejövő levelek)*, akkor értelemszerűen a Local Computernek kopogtatnak a 25-ös portján. *(Hogy mi a feladó portszáma? 1024 feletti random port, esetünkben érdektelen.)*
- ha azonban Local gépről megy Remote irányban a levél, akkor pedig a Remote Computer 25-ös portját nyitogatjuk, tehát „rossz" a filterünkben a portmegadás! *(Nem rossz, csak nem enged ki...)*

Ezt a dilemmát egyetlen filterrel fel sem lehet oldani. A háromlábú DMZ-ben publikációnként két filtert kell alkotnunk, mert egyetlen filter nem képes „lefedni" mind a kimenő, mind pedig a bejövő csomagokat.

A fent bemutatott filteren kívül tehát kell alkotnunk egy olyat is, amelyben a Local és Remote füleken ugyanúgy állítunk be mindent, de a portok megadásánál fordítva járunk el.

Ez azt jelenti, hogy minden „publikálandó" szolgáltatáshoz két filtert kell gyártanunk? Igen is, meg nem is. Vegyük például a HTTP esetét. Az esetek elsöprő többségében tökéletesen elegendő a bejövő 80-as porti forgalom engedélyezése, hisz egy webszerver böngészni reménlhetőleg nem akar. Kivéve, ha például System Update Services fut rajta, mert az éjszakánként nekilódul, és „leböngészi" a legújabb javításokat a WindowsUpdate weboldalról. Ilyenkor bizony létre kell hozni egy kiengedő HTTP-filtert is.

A publikus DMZ összehasonlítása a többi modellel

Ne feledjük: a háromlábú modellben routolás történik, igaz, packet filterekkel korlátozott módon. Ha egy filter átenged egy forgalmat, az változtatás nélkül megy át. *(Az intrusion detectorok továbbra is működnek.)*

Ettől a közvetlen hozzáféréstől sokan félni szoktak, pedig véleményem szerint ez alaptalan. A LAT-os DMZ-k IP-cím fordítása semmiféle védelmet nem jelent, hiszen amit beengedünk a DMZ-be, azt beengedjük, bent is van. Tehát a címfordítás elmaradása sem növeli a kockázatot: akár routolt, akár NAT-olt az elérés, egy jó kis buffer overrun simán eléri a kiszolgálónkat. Van azonban egy lényeges hátránya a háromlábú ISA-nak, mégpedig a HTTP-forgalom tekintetében.

A háromlábú ISA-modellben a HTTP protokoll kezelése némiképp kínos, mert:

1. a DMZ-be tartó HTTP-kérés nem megy át a Web Proxy szolgáltatáson, így sem SSL-láncolás, sem furfangos átirányítások, sem Reverse Proxy nem valósítható meg a „publikált" gépekre
2. nincs tehát HTTP-forgalom ellenőrzés, mert nincs külön HTTP-betörésérzékelő sem. Ezt a funkciót is a Web Proxy látná el.

Ezt akár ki is próbálhatjuk. Ha betelnetelünk a 80-as porton egy Web Publishinggel publikált kiszolgálóra, és a megjelenő ablakba beírunk egy random HTTP-„parancsot" *(például „sdfgkswut" :-)*, az nem jut el a webszerverig, mert az ISA Server elküld minket a pokolba. Ha viszont a harmadik lábon lógó ál-publikált HTTP-kiszolgálóval tesszük ugyanezt, a kérés odaér, és a webszerver küld el minket melegebb éghajlatra – nem pedig a tűzfal.

Ez az ára az olcsó biztonságnak. Mindenki döntse el maga, melyiket választja: a kiszolgálók belső hálózatra helyezését *(Web Proxy rulez!)*, vagy azok külső DMZ-be helyezését, amivel elkerülheti, hogy bármit is elérhetővé kelljen tenni a belső hálózaton.

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

Kapcsolódó tanfolyamaink:
2159 – ISA Server

Biztonságos aláírás kezelő alkalmazás készítése V.

Az aláírás-ellenőrző alkalmazás

Az előző részben az aláírás kezelő alkalmazás adatai, valamint az aláírás létrehozó alkalmazás ezeket hasznosító komponensei, s az ezekkel szemben támasztott követelmények lettek ismertetve. Most az aláírás-ellenőrző alkalmazást mutatjuk be hasonlóképpen. Ezzel nagy ívű sorozatunk egyben a végére is ér. Bár nem fért bele részletesen minden, a lényeget sikerült belegyömöszölni. Aki a leírtakat magáévá tudta tenni bátran nekivághat egy aláírás kezelő alkalmazás fejlesztésének vagy bevezetésének.

Az ellenőrzés feltételei

Az aláírás ellenőrzéséhez a következők szükségesek:

- a nyílt, aláírással ellátott üzenet;
- az aláírásnak a kezdeti ellenőrzéshez szükséges minimális elemeket *(ES)* tartalmazni kell;
- a megfelelő érvényességi adatoknak rendelkezésre kell állni, illetve elérhetőnek kell lenni;
- egy megbízható ellenőrző rendszer az ellenőrzés elvégzésére.

Kezdeti és utólagos *(rendszeres)* ellenőrzés

Az aláírás ellenőrzésének két alapvető típusát különböztetjük meg:

A kezdeti ellenőrzés egy ellenőrző által végrehajtott folyamat, melyet egy aláírás létrehozása és ellenőrzőhöz juttatása után minél hamarabb meg kell tenni. A kezdeti ellenőrzés célja, hogy azok az információk megszerezhetők legyenek, melyek az ellenőrzőnek szükségesek és utólagos ellenőrzésekhez is alkalmassá teszik az aláírást. Ez gyakorlatilag az aláírás első ellenőrzését jelent, mely egy on-line tranzakciós rendszer esetében az aláírást követően pár másodpercen vagy percen belül, egy email esetében tipikusan egy-két órán belül következik be. Ekkor történik az aláíró tanúsítványának, a tanúsítási lánc tanúsítványainak, a tanúsítványokhoz tartozó visszavonási listáknak és egyéb érvényességi adatoknak a beszerzése (*tipikusan az aláírótól vagy a hitelesítési szolgáltatótól*).

Az utólagos *(rendszeres)* ellenőrzés egy ellenőrző *(vagy döntőbíró)* által végrehajtott folyamat, melyet hosszú idővel az aláírás létrehozása után is végre lehet hajtani, s amelyhez nincs szükség több adat megszerzésére, mint amit a kezdeti ellenőrzés során már megszereztek. Ez alól az a helyzet csak a kivétel, mikor az évekkel korábban megtett aláírás biztonsága a közeljövőben kétségessé válik *(a kriptográfiai eljárások elöregedése miatt)*, és ezért további információk beszerzése is szükséges lehet. A kezdeti ellenőrzésnek az érvényesítési adatokat el kell tárolnia, úgy hogy azok az utólagos ellenőrzés számára elérhetők legyenek.

Nem gyakori probléma, de előfordulhat, hogy a kezdeti ellenőrzés túl későn következik be, s így az aláíráskor érvényes ellenőrzési adatok már nem hozzáférhetők. Ez jellemzően a visszavonási lista periodikus kibocsátásból következik be *(minél sűrűbb a kibocsátás, annál inkább előfordulhat ez)*, valamint abból, hogy a lejárt tanúsítványok kikerülhetnek a tanúsítványtárból. Az aláírási szabályzatnak ezekre az esetekre megfelelő intézkedések életbe léptetésével gondoni kell.

Aláírás idejének megállapítása

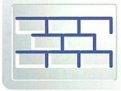
Az aláírás idejének megállapítása a letagadhatatlanság biztosítása és az aláírás érvényességének helyes megállapítása végett perdöntő jelentőségű. Amennyiben az aláírás időbélyegzővel vagy időjelzéssel ellátott *(ES-T)*, akkor ez nem jelent problémát, ellenkező esetben más módszert kell találni aláírás idejének meghatározására.

> A kezdeti ellenőrzés során az ellenőrzőnek is érdemes lehet időbélyegzést kérni az aláírásra és az ellenőrző adatokra. Ezzel később igazolhatja az ellenőrzés eredményét.

A kezdeti ellenőrzés során az ellenőrző élhet például azzal a feltételezéssel *(amennyiben a használt tranzakciós környezet működési logikája erre utal, és az aláírói dokumentumból, az aláírásjellemzőkből, illetve egyéb adatból más nem következik)*, hogy az aláírás ideje az ellenőrzést nem sokkal előzte meg, s így az aláírás idejét behelyettesítheti az ellenőrzés idejével. Ennek annyi hátulütője lehet, hogy egy az aláírás tényleges időpontjában még érvényes aláírás visszautasításra kerül, mert az aláírás és ellenőrzés ideje között vált az aláírás érvénytelenné. Az utólagos *(rendszeres)* ellenőrzés során ez a módszer a kezdeti ellenőrzés idejével alkalmazható, melyet így nyilván tárolni kell az érvényesítési adatok mellet. Ha nem áll rendelkezésre autentikus időinformáció, akkor a kezdeti ellenőrzés során az ellenőrző is kérhet időbélyegzést az aláírásra és az érvényesítési adatokra. Ezzel később igazolhatja, hogy az elfogadás során milyen adatokra támaszkodott.

Egyéb idő megfontolások

Nem csak a túl késői, hanem a túl hamar történő ellenőrzés is problémát jelenthet, sőt ez okoz gyakrabban gondot. Különösen így van ez visszavonási lista használata esetén, mikor az aláírás idejére érvényes lista sokszor csak az aláírást és el-



lenőrzést követően órákkal kerül kibocsátásra. Az ellenőrző számára így rendelkezésére álló lista emiatt még nyilván nem tartalmazhatja azt a tanúsítványt, melynek visszavonására az aláírást közvetlenül megelőzően vagy azt követően került sor. Az aláírás ellenőrzési szabályzat e probléma kezelésére tartalmazhat egy türelmi időt, ami az aláírás, illetve az ellenőrzés kezdeti időpontjától számított várakozást jelent. Az ellenőrző csak e várakozást követően ellenőrizheti a visszavonási állapotinformációkat. Ennek a türelmi időnek a mértéke tipikusan az állapotinformációk késleltetésének idejéhez igazodik. Visszavonási lista esetén hosszabb, OCSP esetén egy rövidebb időt jelent, s garantálja, hogy az ellenőrző a türelmi idő leteltével a számára releváns állapotinformációkhoz fér hozzá. Így ha a tanúsítvány visszavonására az aláírást megelőzően vagy közvetlenül azt követően kerül sor, garantált az aláírás ellenőrzés ez esetben elvárható „sikertelen" eredménye.

Az aláírási szabályzat tartalmazhat előírást az aláírás és időbélyegzés közötti maximális időtartamra is *(hiszen a két időpont között hosszabb idő is eltelhet, főleg ha az időbélyegzést az ellenőrző végzi)*. Minél hosszabb ez az idő, annál nagyobb a lehetőség arra, hogy a két időpont között az aláírás érvénytelenné válik a magánkulcs visszavonása miatt.

Aláírás érvényesítési adatok

Az aláírás ellenőrzése során a következő adatok kerülnek felhasználásra:

- Tanúsítványok *(az aláíró tanúsítványa, attribútum tanúsítványa, és szolgáltatói tanúsítványok)*
- Visszavonási állapotinformációk *(visszavonási listák, OCSP válaszok)*
- Időbélyegzők és időjelzések

Az időbélyegzésre az idők folyamán többször is szükség lehet, amennyiben az előzőleg használt időbélyegző algoritmusa gyengévé válna. Ilyen esetben az új időbélyegző felülbélyegezheti az előző időbélyegzőt, s így egymásba ágyazott időbélyegzők alakulhatnak ki.

Az ellenőrzési folyamat eredményei

A kezdeti és utólagos aláírás ellenőrzési folyamatoknak egyaránt eredménye egy kimeneti állapot. Ennek értéke kezdeti ellenőrzés esetén a következők valamelyike lehet: befejezetlen ellenőrzés, sikeres ellenőrzés, sikertelen ellenőrzés. Utólagos ellenőrzés esetén: sikeres ellenőrzés és sikertelen ellenőrzés.

A sikeres ellenőrzés azt jelenti, hogy az aláírás ellenőrzése, megfelelve az aláírás ellenőrzési szabályzatnak, sikeres volt.

A sikertelen ellenőrzés azt jelenti, hogy az aláírás nem felel meg az aláírás ellenőrzési szabályzatnak, mert például az aláírás formátuma vagy a digitális aláírás értéke nem megfelelő, vagy pedig az aláíró tanúsítványát visszavonták.

A befejezetlen ellenőrzés nem jelenti azt, hogy az aláírás-ellenőrzési folyamat sikertelen volna. A folyamat akkor szolgáltat ilyen eredményt, ha mind az aláírás formátuma, mind a digitális aláírás ellenőrzése sikeres volt, de mégsem áll rendelkezésre elegendő információ ahhoz, hogy az aláírási szabályzat alapján az aláírás érvényességét egyértelműen meg lehessen határozni. Ennek oka – többek között – lehet az, hogy az aláírási szabályzat szerint meg kell várni az aláírás időpontjára érvényes visszavonási lista közzétételét, vagy a tanúsítvány éppen felfüggesztett állapotban van, s meg kell várni, míg érvénytelen vagy újra érvényes lesz. Ekkor az aláírás egy későbbi időpontban való újbóli ellenőrzése kérvényezhetővé tehető, amikor a *(még szükséges)* kiegészítő érvényesítő információ már rendelkezésre fog állni. Befejezetlen ellenőrzés esetében az alkalmazás vagy a felhasználó számára olyan információkat lehet hozzáférhetővé tenni, melyek segítségével az alkalmazás vagy a felhasználó el tudja dönteni, hogy mit kezdjen a részben érvényes elektronikus aláírással.

A kezdeti aláírás ellenőrzési folyamatnak a kimeneti állapoton kívül eredménye az érvényesítési adatok is. Az érvényesítési adatokat az ellenőrzőnek kell összegyűjteni, de az aláíró is biztosíthatja egy részüket vagy egészüket *(pl. ES-X)*.

Specifikus és dinamikus aláírási szabályzat ellenőrzése

Az aláírás-ellenőrzési rendszernek az aláírás aláírási szabályzatnak megfelelő ellenőrzéséhez képesnek kell lenni a szabályzat feldolgozására. Az aláírás-ellenőrzési rendszernek két fő típusa létezik a szerint, hogy az aláírási szabályzatot miként dolgozza fel.

- Specifikus aláírási szabályzatok ellenőrzésére képes rendszerek: Ezek csak bizonyos *(egy vagy több)* a rendszerbe beépített aláírási szabályzat feldolgozására képesek. E szabályzatok tipikusan nem gépi feldolgozásra készültek, hanem olvasható formában léteznek. Ilyen rendszereket jóval egyszerűbb kifejleszteni, mint a dinamikus ellenőrzésére képes rendszereket, utólagosan azonban körülményes beilleszteni további szabályzatok ellenőrzését. Az ilyen rendszerek tanúsítását a támogatott aláírási szabályzatoknak megfelelően *(azokkal összevetve és tesztelve)* kell elvégezni.
- Dinamikusan programozható aláírási szabályzatok ellenőrzésére képes rendszerek: E rendszerek a megfelelő módon definiált aláírási szabályzatok mindegyikének ellenőrzésére képesek, anélkül, hogy a rendszer utólagos módosítást igényelne. E rugalmasság sem végtelen azonban, rendszerint csak egy adott definíciós rendszer *(nyelve, szintaxisa, szemantikája)* szerint működőképes, annak előre meghatározott ellenőrzési készletének, eljárásainak megfelelően. Ha az aláírás-ellenőrzési rendszer egy általa ismeretlen rendszer szerint definiált aláírási szabályzattal találkozik, azt egyáltalán nem képes feldolgozni. Ha pedig a definíciós rendszer bővül, akkor az aláírás-ellenőrzési rendszer továbbfejlesztésére lehet szükség. Az ilyen rendszerek tanúsítását a támogatott definíciós rendszernek megfelelően *(annak funkciókészletével összevetve és tesztelve)* kell elvégezni. Tipikusan több különböző szabályzattal és aláírással ellenőrizve a rendszer működését.

Ellenőrzési rendszerek típusai

Az aláírás ellenőrzési rendszereknek három fő típusát különböztetjük meg:

- Emberi közreműködéssel működő rendszerek
- Automatikus aláírás-ellenőrző rendszerek
- Harmadik fél által végzett ellenőrzésen alapuló rendszerek

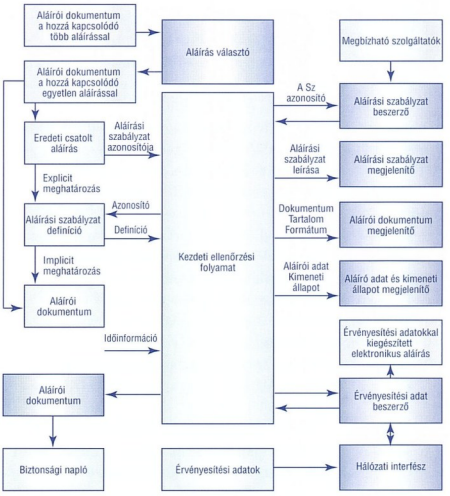
A harmadik fél által végzett ellenőrzésen alapuló rendszerek esetében az alkalmazásnak az ellenőrzést végző harmadik fél

(pl. érvényességi szolgáltató) hitelesítéséről, és a vele való kommunikáció biztonságáról kell gondoskodni.
Az automatikus aláírás-ellenőrző rendszerek emberi közreműködés nélkül működnek, nincs is humán interfészük. Az aláírás ellenőrzési eljárás eredményét és adatait rögzíteniük kell oly módon, hogy az később bármikor értelmezhető lehessen.
Az emberi közreműködéssel működő rendszerek esetében az ellenőrzések egyes részletei automatizáltak, de az emberi közreműködésre és jóváhagyásra a rendszer számít. Az alábbiakban az ilyen rendszerek felépítését és követelményeit mutatjuk be.

Az alkalmazás komponensei

Az aláírás-ellenőrző alkalmazás egy ellenőrző eljárásból, valamint különböző interfészekből áll. Az egyes interfész komponensek a következők:

- **Aláírás választó:** interfész, amin keresztül az ellenőrző megadhatja a feldolgozandó aláírói dokumentumot, s amennyiben ahhoz több aláírás lett csatolva, kiválaszthatja közülük az ellenőrizni kívánt aláírást.
- **Aláírói dokumentum megjelenítő:** feladata, hogy az aláírói dokumentumot a tartalomformátumnak megfelelően (amennyiben ilyen alkalmazásra került) megjelenítse.
- **Aláírási szabályzat beszerző:** az aláírásban explicit definiált aláírási szabályzatok ellenőrző alkalmazás számára történő beszerzéséről gondoskodik, jellemzően egy megbízható szolgáltatón keresztül.
- **Aláírási szabályzat megjelenítő:** opcionális komponens, melynek feladata, hogy az aláírási szabályzatot megjelenítse.
- **Aláíró adat és kimeneti állapot megjelenítő:** az aláírás ellenőrzésének eredményeként megjeleníti az aláíró nevét és az ellenőrzés kimeneti állapotát.
- **Érvényesítési adat beszerző:** Az aláírás kezdeti ellenőrzéséhez szükséges, az aláírásban nem szereplő, további aláírás érvényesítési adatok beszerzéséről gondoskodik.
- **Biztonsági naplózó:** opcionális komponens, ami egy független szolgáltató rendszerébe biztonságos módon rögzíti az aláírás érvényesítési adatait.
- **Hálózati interfész:** komponens, amely az aláíró által nem biztosított aláírás érvényesítési adatok hálózati kapcsolaton keresztül történő beszerzéséhez szükséges *(pl. visszavonási listák, időbélyegzők)*.



Kezdeti aláírás-ellenőrző rendszer

Aláírás választó

Amennyiben több aláírás lett csatolva a feldolgozandó aláíróí dokumentumhoz, akkor az interfész ezen aláírásokat és azok esetleges korábbi ellenőrzésének a kimeneti állapotát (eredményét) kijelzi, s az ellenőrző számára lehetővé teszi az ellenőrizni kívánt aláírás kiválasztását.

Aláíróí dokumentum megjelenítő

Aláíróí dokumentum megjelenítő az aláírt tartalomformátumnak megfelelően megjeleníti az aláíróí dokumentumot az ellenőrző számára. A megjelenítésnek az „Azt lett aláírva, amit látsz" elvet kell követnie. Amennyiben a megjelenítő nem képes az aláíróí dokumentumot ilyen módon megjeleníteni, akkor azt jeleznie kell.

Aláírási szabályzat beszerző

Az aláírásban explicit definiált aláírási szabályzatok, valamint az aláírási szabályzatban hivatkozott, de rajta kívül definiált elemek (pl. gyökér tanúsítványok) beszerzéséről gondoskodik (letölti hálózaton keresztül). Explicit nem definiált aláírási szabályzatok azonosítása nem az aláírás-ellenőrző rendszer feladata. Külső azonosítás után azonban e komponens beszerezheti az ilyen szabályzatokat is.

Aláírási szabályzat megjelenítő

Aláírási szabályzat megjelenítő feladata, hogy az aláírási szabályzatot, illetve annak felhasználási területét, és az aláíráshoz kapcsolódó feltételeket megjelenítse az ellenőrző számára. Kizárólag dinamikus aláírási szabályzatokat feldolgozó rendszerek számára nem szükséges ilyen komponens (hiszen a szabályzatot a rendszer automatizmusa dolgozza fel).

Aláíró adat és kimeneti állapot megjelenítő

Az aláírás ellenőrzésének eredményeként a komponens megjeleníti az aláíró és az aláíróí tanúsítványt hitelesítő hitelesítésszolgáltató nevét. Az aláíró nevét az aláíráshoz csatolt tanúsít-

vány egyedi nevéből veszi. Ha az aláíráshoz az aláíró nem csatolta a tanúsítványát, akkor először az aláírói tanúsítványt hitelesítő hitelesítésszolgáltató nevét jeleníti meg, s ha azt az ellenőrző elfogadja, akkor beszerzi a szolgáltatótól az aláíró tanúsítványt, s ezt követően jeleníti meg az aláíró nevét. Az aláírói tanúsítványt hitelesítő hitelesítésszolgáltató nevének megjelenítésére mindenképp szükség van, hiszen az aláíró neve csak a szolgáltató szabályai szerint és annak névterében értelmezhető.

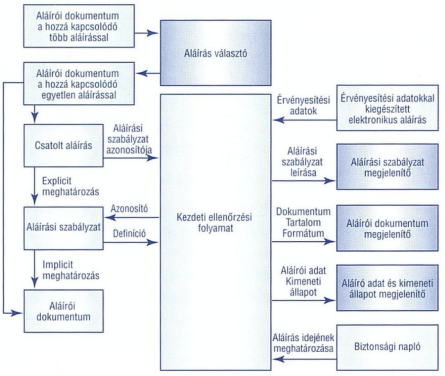
Ezen adatokon kívül az ellenőrző kérésére a következő adatokat jeleníti meg *(amennyiben azok rendelkezésre állnak)*:

- Az elektronikus aláírás időjelzése
- Az időbélyegző adatai
- Az aláírás helye
- A kötelezettségvállalás típusa
- Az aláíró kinyilvánított vagy hitelesített szerepköre
- Az aláíró tanúsítványa

A komponens az aláírási szabályzatnak megfelelően megjeleníti az ellenőrzési eljárás kimeneti állapotát (eredményét) is, ami befejezetlen ellenőrzés, sikeres ellenőrzés és sikertelen ellenőrzés lehet.

Érvényesítési adat beszerző

Amennyiben az aláírás kezdeti ellenőrzése befejezetlen, akkor az aláírás ellenőrzéséhez szükséges további aláírás érvényesítési adatok beszerzésére felhívja az ellenőrző figyelmét. Az így beszerzett adatok ellenőrzési eljárásba való bejuttatásáról és az eljárás befejezéséről *(amennyiben lehetséges)* a komponens egyúttal gondoskodik is.



Utólagos aláírás-ellenőrző rendszer

Általános követelmények

Az aláírás-ellenőrző alkalmazás tervezésekor és fejlesztésekor számos biztonsági és funkcionális követelményt kell a szakembereknek figyelembe venni. A fontosabb és érdekesebb követelményeket, a követelményt indokoló veszélyekkel, és néhol a lehetséges megoldásokkal együtt bemutatom. A biztonsági követelmények között vannak általánosak, melyek az alkalmazás egészére vagy összes moduljára vonatkoznak, valamint az egyes adatokra és komponensekre specifikusak. Az érvényességi adatok ellenőrzési szabályai szintén az alkalmazás követelményei közé tartoznak.

Az általános biztonsági követelmények közé sorolhatóak a következők:

- Az ellenőrzőnek képesnek kell lenni bármilyen az ellenőrző rendszerben történt, annak működését és biztonságát befolyásoló változás észlelésére.
  Azért, hogy az ellenőrző alkalmazást a telepítést megelőzően és azt követően se lehessen észrevétlenül módosítani.
- Az aláírás ellenőrzésének az aláírás ellenőrzési szabályzatnak megfelelően kell történni, mind a specifikus aláírási szabályzatok ellenőrzésére képes rendszerek, mind a dinamikusan programozható aláírási szabályzatok ellenőrzésére képes rendszerek esetében.
- A kezdeti ellenőrzés során az aláírási szabályzatnak megfelelően össze kell gyűjteni az aláírás érvényességének megállapításához szükséges összes érvényesítési adatot.
  Ezek egy részét tartalmazhatja az aláírás maga *(pl. ES, ES-C)*, más részükre hivatkozással élhet *(pl. ES-C)*, vagy az aláírási és más szabályzatból következhet. Ez utóbbiakat külső forrásból kell beszerezni.

Érvényességi adatok ellenőrzési szabályai

- Egyértelműen meg kell határozni, hogy az aláírás pillanatában az aláíró tanúsítványa, attribútum tanúsítványa *(ha ilyen szerepel az aláírandó adatok között)*, és az ezekhez kötődő tanúsítási lánc érvényes volt-e.
  Kizárva az esetlegesen érvénytelen adatokon alapuló aláírásokat.
- Be kell szerezni az aláírás érvényességét igazoló teljes tanúsítási láncot, és annak minden tanúsítványához kötődően azok visszavonási állapotinformációit *(visszavonási listákat, OCSP válaszokat)*.
- Az érvényesítési adatok aláírás, illetve ellenőrzéskori valódiságának igazolása végett be kell szerezni egy időbélyegzőt vagy időjelzést az aláírásra *(és opcionálisan az érvényesítési adatokra)* vonatkozóan.
  Az időbélyegzőt tartalmazhatja az aláírás maga *(pl. ES-T)*, ellenkező esetben az ellenőrzőnek kell beszerezni.
- Minősített aláírások ellenőrzésekor meg kell győződni arról, hogy az aláíró tanúsítványa biztonságos aláíróeszközt alkalmazó minősített tanúsítvány.
  Az RFC 2459 szabvány szerint definiált Certificate Policies toldaléknak az ETSI 101-456 által meghatározott QCP public + SSCD értékűnek kell lennie (0.4.0.1456.1.1). A Key Usage toldaléknak jelen kell lenni, Non-Repudiation bitjének beállítva, Digital Signature bitjének nem beállítva kell lennie.
- Az aláírás érvényességének megállapításához fel kell építeni egy tanúsítási láncot az aláíró tanúsítványát kibocsátó hitelesítő egység és az aláírási szabályzat által elismert bizalmi pontok között.
- Az aláírás érvényességének megállapításához a tanúsítási lánc összes tanúsítványának (beleértve a bizalmi pont és a végfelhasználó tanúsítványát is) megkötéseit figyelembe kell venni.
  E megkötéseket a Basic Constraints, a Name Constraints, a Policy Constraints és a Policy Mappings toldalékok írják le.

Aláírás választó komponens

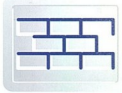
- Lehetővé kell tenni az ellenőrző számára, hogy megadja a feldolgozandó aláírói dokumentumot, s amennyiben több aláírás lett csatolva hozzá, kiválaszthassa az ellenőrizni kívánt aláírást.
  Ehhez az aláíráshoz kapcsolódó aláíró nevek *(tanúsítvány egyedi név)* kijelzése szükséges.

Aláírói dokumentum megjelenítő komponens

- Az aláírói dokumentum megjelenítőnek garantálni kell, hogy az ellenőrző számára egyértelmű, biztonságos módon, az aláíró által meghatározott tartalomformátumnak megfelelően történik az aláírói dokumentum megjelenítése.
  Ne fordulhasson elő, hogy az ellenőrző az aláírói dokumentumot rosszul értelmezi, mert az nem megfelelő módon jelenik meg.
- Amennyiben aláírói dokumentum megjelenítő nem képes az aláírói dokumentum tartalomformátumnak megfelelő módon történő megjelenítésére, akkor azt jeleznie kell az ellenőrző számára.

Aláírási szabályzat beszerző komponens

- A kezdeti ellenőrzés során egyértelműen azonosítani kell az aláírással kapcsolatos aláírási szabályzatot, akár explicit (az aláírásban) vagy implicit (az aláírói dokumentumban) történik rá hivatkozás.
  Az implicit hivatkozás feloldásához szükséges lehet humán közreműködés.
- Az ellenőrző számára biztonságos módon biztosítani kell a kérdéses aláírásra vonatkozó aláírási szabályzat másolatát.
  Jellemzően megbízható szolgáltatótól való letöltés által.
- Meg kell győződni arról, hogy az aláírási szabályzat valóban az adott aláíráshoz tartozik.
  Össze kell vetni az aláírásban jelzett aláírási szabályzat azonosítót *(OID)* és lenyomatot a beszerzett aláírási szabályzat tényleges azonosítójával és lenyomatával.
- Garantálni kell az aláírási szabályzat hitelességét és sértetlenségét.
  Ehhez ellenőrizni kell az aláírási szabályzaton szereplő aláírást.
- Amennyiben az aláírási szabályzat külsőleg definiált elemekre hivatkozik, akkor garantálni kell ezen elemek hitelességét és sértetlenségét.
  Ellenőrizni kell az elemeken szereplő aláírást, vagy más, hasonló biztonságot garantáló megoldást kell alkalmazni.



Aláírási szabályzat megjelenítő komponens

- Aláírási szabályzat megjelenítőnek garantálni kell, hogy az ellenőrző számára egyértelmű, biztonságos módon történik az aláírási szabályzat azonosítójának, felhasználási területének és egyéb az aláírásra vonatkozó feltételeinek megjelenítése.
  A beavatkozás kizárására és a tartalomformátum felismerésére ez esetben is szükség van.
- Amennyiben az aláírási szabályzatra implicit történik hivatkozás, képesnek kell lenni biztonságosan megjeleníteni azt a dokumentumot, mely a kérdéses aláírásra vonatkozó aláírási szabályzatra implicit hivatkozik. Szerződés, jogszabály vagy egyéb szabályzat lehet ilyen dokumentum.

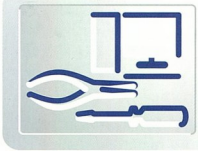
Aláíró adat és kimeneti állapot megjelenítő

- Az aláíró adat és kimeneti állapot megjelenítőnek garantálni kell, hogy az ellenőrző számára egyértelmű és biztonságos módon történik az aláíró és hitelesítésszolgáltató nevének, az aláírás egyéb adatainak, valamint az ellenőrzés kimeneti állapotának a megjelenítése.
  A folyamatba való esetleges beavatkozások kizárásával, a névformátum helyes értelmezésével, a használt karakterkészlet felismerésével és kezelésével kell ezt megtenni.

Érvényesítési adat beszerző komponens

- Érvényesítési adat beszerzőnek a kezdeti ellenőrzés során lehetővé kell tenni az ellenőrző számára az aláírás ellenőrzéséhez szükséges további aláírás érvényesítési adatok beszerzését, valamint gondoskodnia kell az ellenőrzés megfelelő kimeneti állapotáról.
  Ennek érdekében figyelmeztetnie kell az ellenőrzőt arra, hogy mi hiányzik, és az esetlegesen beszerzett adatok feldolgozását lehetővé kell tennie.

Almási János
CIO
Almasi-Janos@dbrt.hu



A tranzakciónapló és környéke

SQL Server 2000 adat- és tranzakciónapló-fájlok, adatbázis helyreállítási modellek, mentési stratégiák

Az adatok rendszeres mentésének fontosságát mindenki tudja. Arról már olykor elfeledkezünk, hogy felmérjük, mennyi tranzakció (új adat, vagy módosítás) elvesztését tudjuk elviselni. Az SQL Server tranzakciónaplóját sok rendszergazda „nem szereti". Gyakori kérdés, miért nem lehet a tranzakciónaplót kikapcsolni? Meglepően sok adatbázis „simple" helyreállítási modellben működik, ami a tranzakciónapló automatikus ürítésével jár. Ez (többnyire) megakadályozza, hogy a logfájl nagyra nőjön, de ugyanakkor lemezhiba, vagy egy véletlen táblatörlés után lehetetlenné teszi az utolsó pillanatig érvényes helyreállítást.

Az SQL Server 2000 *(és 7.0)* adatbázisok legalább egy adat- és legalább egy tranzakciónapló-fájlból állnak. Az első adatfájl a Master Data File, ezért ezt .mdf kiterjesztéssel szokás jelölni. A további adatfájlok kiterjesztése szokás szerint .ndf, a tranzakciónapló-, vagy logfájloké pedig .ldf. Ezek a jelölések csak konvenciók, használatuk nem kötelező. Az adatfájlok fájlcsoportokba szerveződnek. Az .mdf fájl kötelezően a „PRIMARY" fájlcsoportba tartozik, az .ndf fájlok különböző csoportokba kerülhetnek. A kisebb adatbázisok többnyire egyetlen .mdf és egy .ldf fájlból állnak. Nagyobb adatbázisok esetén gyakori több másodlagos adatfájl *(.ndf)* használata, de több tranzakciónapló-fájlra ritkán van szükség. *(Kivétel:* ***[Q328551]****)*

A tranzakciónapló szerepe

A tranzakcióktól elvárt tulajdonságok az atomiság, konzisztencia, izoláció és tartósság; angol rövidítéssel: ACID. Az elvárt tranzakciótulajdonságok közül három *(atomi, konzisztens és tartós)* a tranzakciónapló segítségével valósul meg.

A legtöbb modern adatbáziskezelőhöz hasonlóan az SQL Server ún. előre írt *(write-ahead)* naplózást végez. Minden adatmódosítás először a tranzakciónaplóba íródik, és csak ezután az adatlapra. A tranzakció érvényesítése *(commit)* alkalmával a log mindenképpen diszkre íródik. Ettől a pillanattól kezdve a tranzakció „tartóssá" válik: ha bármilyen ok, például hálózat-kimaradás miatt a módosított adatlap nem íródik vissza a diszkre, a tranzakciónapló alapján a módosítást az SQL Server „előregörgeti" a legközelebbi újraindításkor. Azok a tranzakciók, amelyek nem zárultak le az SQL Server leállítása, illetve a hálózatkimaradás időpontjáig, visszagörgetődnek *(rollback)*. A **[Q230785]** cikk egy tranzakció lefolyását a következő, egyszerűsített modellel érzékelteti:

```
BEGIN TRANSACTION
   INSERT INTO tblTest VALUES (1)
COMMIT TRANSACTION
```

Utasítás	Tevékenység
BEGIN TRANSACTION	BEGIN TRANSACTION bejegyzés írása a log gyorsítótárba *(cache-be)*, legalábbis elméletileg. A gyakorlatban az SQL Server a tranzakciót csak az első adatmódosítás pillanatában tekinti megkezdettnek, tehát a BEGIN TRANSACTION bejegyzés is csak az INSERT megkezdésekor kerül a naplóba.
INSERT INTO tblTest VALUES *(1)*	• A tábla megfelelő adatlapját az SQL Server beolvassa az adat gyorsítótárba *(ha még nincs ott)*. • A lapot egy „latch" gyors szinkronizációs zárral védi, megtiltja a diszkre írását, módosított *(dirty)* lapként jelöli meg és megszerzi a megfelelő tranzakcionális zárakat. • Egy INSERT bejegyzés kerül a log gyorsítótárba. *(DELETE utasítás esetén a törölt sor, UPDATE esetén a módosítás előtti és utáni állapot is naplózódik.)* • Az új adatsor felíródik az adatlapra. • A latch felszabadul. A log megfelelő bejegyzéseit még nem szükséges diszkre írni, hiszen egyelőre minden módosítás csak a memóriában történt.
COMMIT TRANSACTION	• Egy COMMIT bejegyzés készül és a tranzakciónaplónak a tranzakcióra vonatkozó bejegyzései a diszkre íródnak. A tranzakció csak akkor tekinthető érvényesítettnek, ha a log megfelelő bejegyzései diszkre, azaz nem törlődő tárolóra íródtak. A módosított adatlap a gyorsítótárban

	marad. Ha ekkor következne be egy hálózat-kimaradás, a tranzakciónapló alapján az SQL Server rekonstruálni tudja a módosításokat. • A tranzakcionális zárak felszabadulnak.
CHECKPOINT	A beállított „recovery interval" konfigurációs paraméter és a módosított adatlapok számának függvényében az SQL Server a módosított adatlapokat időnként diszkre írja. A checkpoint megtörténte is naplózódik. Így újraindításkor az SQL Server tudja, hogy a checkpoint bejegyzés előtt lezárult tranzakciók módosításai már mindenképpen a diszken vannak, ezeket nem szükséges előregörgetni, így az adatbázis helyreállítása gyorsabb lesz.

Az adatbázis checkpoint és a log

Adatbázis checkpoint történik a CHECKPOINT parancs hatására, az adatbázis opciók megváltoztatása miatt, az SQL Server leállításakor és automatikusan, ha az SQL Server úgy találja, túl sok módosított adatlap van a gyorsítótárban.

A checkpoint az adatbázison a következőket hajtja végre:

- Bejegyzi a tranzakciónaplóba a checkpoint kezdetét és sok egyéb információt.

Az egyik ilyen feljegyzett adat a Minimum LSN. Az LSN a Log Sequence Number rövidítése; egy növekvő sorszám, amely a logbejegyzéseket azonosítja. A Minimum LSN a folyamatban levő tranzakciók visszagörgetéséhez szükséges legelső logbejegyzés, vagy a legrégebbi replikálásra váró bejegyzés azonosítója. Pontosabban, a Minimum LSN az alábbiak közül a legkisebb:

- A checkpoint kezdetének LSN-je.
- A legöregebb nyitott tranzakció kezdetének LSN-je.
- Tranzakcionális replikáció esetén, a legrégebbi, még nem replikált tranzakció bejegyzés LSN-je.

- Ha az adatbázis SIMPLE helyreállítási modellben van, a checkpoint törli a tranzakciónaplónak a Minimum LSN előtti sorait.
- Lemezre írja a módosított adatlapokat a gyorsítótárból.
- Végül a checkpoint bejegyzi a tranzakciónaplóba saját befejezését.

A tranzakciónaplónak a Minimum LSN-től kezdődő és a legfrissebb bejegyzésig tartó részét aktívnak nevezzük. A log aktív része nem törölhető. Ha egy adatbázisban hosszan futó – vagy programozói hiba folytán lezáratlan – tranzakciók vannak, a tranzakciónaplóból nem tudjuk eltávolítani a Minimum LSN utáni bejegyzéseket, vagyis a log egyre csak növekszik. SQL Server 2000 esetén a tranzakciónapló bejegyzéseit a nem dokumentált fn_dblog() függvénnyel tudjuk lekérdezni. *(A nem dokumentált parancsok, függvények, eljárások használata általában nem javasolt, mivel verzióról verzióra megváltozhatnak, vagy megszűnhetnek. Alkalmazást nem célszerű nem dokumentált parancsokra építeni. Információszerzésre viszont használhatjuk őket.)*

Az aktuális adatbázis teljes tranzakciónaplóját a következő parancs listázza ki:

```
select * from ::fn_dblog(NULL, NULL)
```

Az fn_dblog két paramétere az a két LSN, amelyek közé eső bejegyzésekre kíváncsiak vagyunk. Kicsit kényelmetlen, hogy az előbbi lekérdezés eredménye az LSN-eket hexadecimális formában jeleníti meg, de az fn_dblog integer paramétereket vár. *(Például, a '00000042:000001d3:0001' LSN-t '66:467:1' formában kell megadnunk.)*

A virtuális logfájlok

A tranzakciónapló logikailag a logrekordok folyamatos sorozata. Azonban a hatékonyabb logkezelés érdekében az SQL Server a logfájlt több virtuális logfájlra *(VLF-re)* bontja. A VLF-ek mérete és száma változó: az SQL Server dinamikusan alakítja ki őket, amikor egy adatbázist létrehozunk, vagy módosítjuk a log méretét, vagy annak mérete automatikusan megnövekszik.

Ha egy adatbázis logját nagyon kicsire méretezzük, és a log fájl automatikus növelésére kis lépéseket írunk elő, sok-sok VLF keletkezik. Ez egy idő után lassíthatja az adatbázis helyreállítását. Általában jó gyakorlat a tranzakciónapló méretét eleve az adatfájlok méretének húsz százalékára beállítani, így elkerülhető a sok automatikus bővítés, ami önmagában is lassulást okozhat. A húsz százalék persze nem minden esetben jó érték: egy nagyon nagy adatbázis esetén, ahol az adatoknak csak egy százaléka módosul naponta, felesleges nagy tranzakciónaplót készíteni; egy aprócska adatbázis, ahol a teljes adatállomány folyamatosan módosul, esetleg nagyobb tranzakciónaplót igényel, mint az adatfájlok mérete.

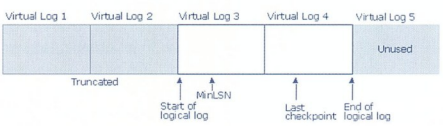
A virtuális logfájlokról a *(nem dokumentált)* dbcc loginfo paranccsal kaphatunk információt. Például:

```
dbcc loginfo('test')
```

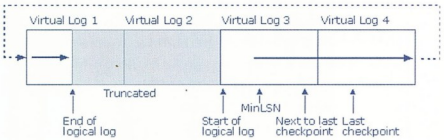
	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	2	253952	8192	9	2	64	0
2	2	253952	262144	0	0	0	0
3	2	270336	516096	10	2	64	9000000042600071
4	2	262144	786432	11	2	64	10000000037200082

Négy sort kaptunk, ennyi virtuális logfájlunk van. A FileSize oszlop mutatja a VLF méretét. A VLF-ek közül jelenleg az van használatban, ahol a Status oszlop tartalma 2.

A fenti lista már sejteti, hogy az SQL Server a VLF-eket körbe láncolva használja. Amikor egy adatbázist létrehozunk, a fizikailag első VLF van használatban. Idővel újabb VLF-ek is elhasználódnak a bejegyzések tárolásához. Előbb-utóbb megtörténik a log csonkolása is – például egy BACKUP LOG parancs hatására. A log csonkolása mindig a Minimal LSN-ig történik, tehát az előtte levő terület felszabadul. A következő ábra egy olyan tranzakciónaplót mutat, ahol a harmadik VLF tartalmazza a MinLSN-t és a negyedik az utolsó naplóbejegyzést. A szürke terület szabad.



Amikor a logikai log eléri a fizikai log fájl végét, az újabb bejegyzések az első szabad VLF-be kerülnek:



Ha a logfájl megtelik, és az automatikus bővítés opció be van kapcsolva, az előírt mértékben növekszik. A növekmény egy, vagy több új VLF-ben jelenik meg.

A log méretének csökkentése

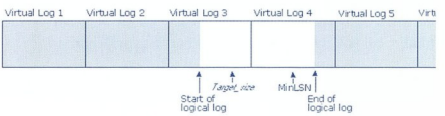
Mikor lehet szükség a log méretének csökkentésére? A jellemző okok:

- Az adatbázisunk FULL, vagy BULK_LOGGED helyreállítási modellben van, és rendszeresen mentjük a teljes adatbázist, de nem mentjük a tranzakciónaplót. Tévhit, hogy a teljes adatbázis mentése csonkolja a tranzakciónaplót.
- Egy szokatlanul nagy *(sok módosítást tartalmazó)* tranzakció miatt a log mérete célszerűtlenül nagy lesz. Például, egy 100 GB-os tábla minden sorának módosítása egyetlen UPDATE utasítással kb. 200 GB logterületet igényel. Egy módosító utasítás akkor is *(elemi)* tranzakció, ha nem határolja BEGIN TRAN és COMMIT. Egy élő, lezáratlan tranzakció log bejegyzései még SIMPLE helyreállítási modellben sem törölhetők, tehát a tranzakciónaplónk legalább 200 GB lesz.
- Egy tranzakcionális replikációban a publikáló adatbázis tranzakciónaplója megnövekszik, mert a log reader valamilyen okból nem dolgozza fel a replikálandó tranzakciókat. *(Például, nem fut az SQL Agent.)*
- Sok, apró lépésben történő automatikus lognövekedés miatt túl sok *(több száz)* VLF keletkezik. Szeretnénk egy hasonló méretű, de kevesebb VLF-ből álló tranzakciónaplót készíteni. Ehhez először össze kell húznunk a logfájlt, majd egy ALTER DATABASE paranccsal, egy lépésben megnövelni.

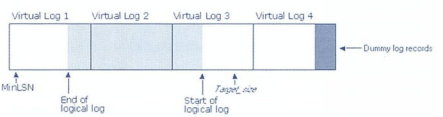
A tranzakciónapló méretét a DBCC SHRINKFILE paranccsal csökkenthetjük. Kisméretű és csak néhány türelmes felhasználó által használt adatbázisok esetén használható az AUTOSHRINK adatbázis opció, ami automatikusan összehúzza az adatbázis fájlok, köztük a tranzakciónapló méretét. Ugyanakkor az automatikus méretcsökkentés előbb-utóbb automatikus méretnöveléshez vezet. Az automatikus méretnövelés munkával jár, tehát lassítja az őt kiváltó tranzakció és a többi, éppen végrehajtás alatt álló tranzakció végrehajtását. Ha jó teljesítménnyel működő SQL Server alkalmazást szeretnénk, legjobb elfelejteni az AUTOSHRINK opciót.

A DBCC SHRINKFILE csak a log fizikai végén levő, üres VLF-eket tudja levágni. Tehát a log méretének csökkentése előtt célszerű a BACKUP LOG paranccsal a logot üríteni. Még ekkor is előfordulhat, hogy a körbeírt log aktív része éppen a fizikailag utolsó VLF-re esik.
Ilyen esetben, az SQL Server 7.0 esetén a DBCC SHRINKFILE látszólag hatástalan. Valójában az SQL Server megjegyzi a log összehúzására vonatkozó utasítást, és amikor a log aktív része elmozdul a logfájl fizikai végéről, újra megkísérli a log kívánt méretre csökkentését. Ha nem akarunk várni, a DBCC LOGINFO paranccsal megvizsgálhatjuk, hogy a log mely részei aktívak, majd mesterségesen generált tranzakciókkal betelítjük az aktuálisan használt virtuális logfájlt. Amikor egy másik VLF-re kerül a log aktív része, újabb BACKUP LOG után a DBCC SHRINKFILE eredményes lesz.
Az SQL Server 2000 esetén a DBCC SHRINKFILE megkönnyíti a dolgunkat. Ha az általunk előírt logméretet nem tudja elérni, mert a log aktív része a fájl fizikai végén van, automatikusan feltölti az aktuális VLF-et, figyelmeztet bennünket, hogy ürítsük *(mentsük)* a tranzakciónaplót. A BACKUP LOG után egy újabb DBCC SHRINKFILE paranccsal tovább tudjuk csökkenteni a log méretét.

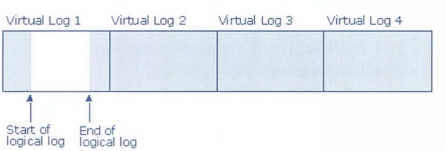
A következő példa egy olyan logot mutat, amely hat, 100 MB-os VLF-ből áll. Jelenleg a VLF3 és a VLF4 van használatban. A DBCC SHRINKFILE paranccsal megadott fájlméret, a Target_size 275MB.



A DBCC SHRINKFILE*(LOGIKAI_FÁJLNÉV, 275)* parancs eltünteti a fájl végén levő üres VLF-eket és a VLF4-et feltölti.



Egy BACKUP LOG parancs után a log aktív része valószínűleg a VLF1-re kerül. *(Ha hosszan futó tranzakcióink vannak, előfordulhat, hogy várnunk kell, mert a lezáratlan tranzakciók bejegyzései nem törölhetők.)*



Ezután egy újabb DBCC SHRINKFILE már akár 200 MB-ra is össze tudja húzni a log méretét. Ennél lejjebb nem tudunk menni, mert a log legalább két VLF-ből áll.

Adatbázishelyreállítási modellek

Egy SQL Server 2000 adatbázis FULL, BULK_LOGGED, vagy SIMPLE helyreállítási *(recovery)* modellben lehet. *(Az SQL Server 7.0 esetén ezek a modellek nem léteztek, illetve a „select into/bulkcopy" és a „trunc. log on chkpt." adatbázis opciók beállításaival szabályozhattuk az adatbázisok működését a helyreállítás szempontjából.)*
A helyreállítási modellt a CREATE DATABASE és az ALTER DATABASE parancsokkal állíthatjuk be. Az alapértelmezés az

SQL Server Enterprise és Standard változatával létrehozott adatbázisok esetén FULL, a Personal Edition és az MSDE adatbázisai esetén pedig SIMPLE. Érdemes megjegyezni, hogy az adatbázis helyreállítási modellje mentés-helyreállítás, vagy lekapcsolás-felkapcsolás *(detach-attach)* után megmarad, éppúgy, mint az adatbázis opciók. Tehát például, egy MSDE-vel fejlesztett adatbázis, amit később egy SQL Server Standard változaton használunk, megmarad SIMPLE modellben, amíg át nem állítjuk. *(Hasonló meglepetés érhet bennünket az AUTOCLOSE adatbázisopció kapcsán.)*

SIMPLE helyreállítási modell esetén az egyszerű kezelés érdekében lemondunk arról, hogy lemezhiba esetén az adatbázist a hiba pillanatáig helyreállítsuk. SIMPLE helyreállítási modellben levő adatbázisról készíthetünk teljes adatbázismentést, fájl és fájlcsoport mentéseket, differenciális mentést, de nem tudjuk a tranzakciónaplót menteni. A tranzakciónapló mentésnek nem is lenne értelme, mivel a checkpoint automatikusan törli a log inaktív részét.
A SIMPLE modell viselkedése közelítőleg megfelel a „select into/bulkcopy" és a „trunc. log on chkpt." adatbázis opciók bekapcsolásának az SQL Server 7.0 esetén.

FULL helyreállítási modellben minden adatbázisművelet teljesen naplózott, még az olyan, nagy tömegű adatot érintő *(bulk)* műveletek is, mint a SELECT INTO, a CREATE INDEX és a BULK INSERT.

FULL helyreállítási modellben levő adatbázis esetén menthetjük a teljes adatbázist, az egyes fájlokat, vagy fájlcsoportokat és a tranzakciónaplót. Az adatbázisról és a fájlokról készíthetünk differenciális mentést is, amely az utolsó „rendes" mentés óta bekövetkezett változásokat tartalmazza. Érdemes megjegyezni, hogy amíg az adatokról nem készítünk mentést, az SQL Server a tranzakciónaplót ugyanúgy automatikusan üríti, mint a SIMPLE modell esetén. A log mentésének csak akkor van értelme, ha előzőleg adatbázist, vagy adatfájlokat mentettünk.

Egy gyakori, jól használható mentési séma a következő:
- Naponta mentjük az adatbázist *(backup database)*;
- Óránként mentjük a tranzakciónaplót (backup log).

Ha bármilyen ok miatt elveszik az adatbázisunk, de a tranzakciónaplót tartalmazó lemez olvasható, a logot még tudjuk menteni, így a hiba bekövetkeztéig tudunk helyreállítást végezni. A tranzakciónaplót célszerű védeni a diszkhibák ellen, például tükrözéssel *(RAID1, vagy RAID10)*. Nem jó ötlet a logot RAID5-re tenni, mert lassú.

A helyreállítás menete:
- Betöltjük az utolsó teljes adatbázismentést *(RESTORE DATABASE ... WITH NORECOVERY)*;
- Sorra betöltjük azokat a tranzakciónapló mentéseket, amelyek az adatbázismentés után készültek *(RESTORE LOG ... WITH NORECOVERY)*;
- Végül betöltjük az utoljára mentett log-ot *(RESTORE LOG ... WITH RECOVERY)*.

Természetesen, a tranzakciónapló mentések sora nem szakadhat meg. Ha elveszítünk egy logmentést, nem tudjuk a következőt sem betölteni. Ha mentés nélkül csonkoljuk a log-ot *(BACKUP LOG WITH TRUNCATE_ONLY)*, akkor már nem is tudunk újabb logmentést készíteni, amíg nem végzünk egy adatbázis- vagy fájlmentést.

Ha egy adatbázismentést veszítünk el, például szalaghiba miatt, még mindig helyre tudjuk állítani az adatbázisunkat, ha megvan valamelyik régebbi adatbázismentés és az utána következő logmentések *(megszakítás nélküli)* sora.

A **BULK_LOGGED** modellben egyes műveletek „gyengén" naplózottak. Ezeknél csak az extent *(8 lapból álló egység)* allokációkat naplózza az SQL Server. Gyengén naplózott művelet a SELECT INTO, a bcp és a BULK INSERT, a CREATE INDEX és a text és image módosítások *(WRITETEXT, UPDATETEXT)*. A gyengén naplózott műveletek előnye, hogy lényegesen gyorsabbak, mintha teljesen naplóznánk őket.

A gyengén naplózott műveletek ellenére a BULK_LOGGED modell megengedi a logmentéseket. Az SQL Server BULK_LOGGED modell esetén nyilvántartja a gyengén naplózott műveletek által módosított extenteket, és ezeket a log mentés során szintén menti. Így a FULL modellnél leírt mentési-helyreállítási séma majdnem teljesen használható BULK_LOGGED modellben is. A kivétel a hiba bekövetkezte utáni logmentés. Ekkor ugyanis már nincs meg az adatbázisunk, tehát az SQL Server nem tudja a logmentés részeként a gyengén naplózott műveletekkel módosított extentek mentését is elvégezni. Ha ilyen műveletek történtek az utolsó logmentés és a hiba bekövetkezése közötti időben, nem tudjuk az adatbázist a hiba pillanatáig helyreállítani.

A potenciális veszély ellenére a BULK_LOGGED modell jól használható, és majdnem olyan biztonságos, mint a FULL modell. A BULK_LOGGED és a FULL modellek között szabadon kapcsolgathatunk: ugyanaz a mentési eljárás működik mindkét modell esetén.

A SIMPLE modellre ez már nem igaz, mivel ott nem tudunk tranzakciónaplót menteni. Ha az adatbázist FULL, vagy BULK_LOGGED modellből SIMPLE-be kapcsoljuk, a mentési eljárás logmentései nem futnak le. Kellemetlenebb, hogy a SIMPLE modellből FULL, vagy BULK_LOGGED modellbe történő váltás után először adatmentést *(adatbázis- vagy fájlmentést)* kell készítenünk, és csak ezután folytatódhat a log mentéseket is tartalmazó mentési séma.

Az adatbázismentés algoritmusa

Az SQL Server 7.0 és 2000 ún. fuzzy mentési algoritmust használ. A fuzzy *(„homályos")* jelzőt azért kapta, mert az adatbázis mentése során nem törődik azzal, hogy az adatlapokat a folyamatban levő munkák módosíthatják – tulajdonképpen inkonzisztens adatokat ment. Azonban a mentés végén az SQL Server elmenti a tranzakciónaplónak azt a részét is, amely a mentés ideje alatt futó tranzakciók bejegyzéseit tartalmazza. Így visszatöltéskor a potenciálisan inkonzisztens adatlapokat az elmentett logdarab alapján helyre lehet állítani. Ez ugyanaz a helyreállítási folyamat, ami az SQL Server indításakor minden adatbázisra megtörténik. A fuzzy algoritmus előnye, hogy gyors, mert az adatbázis extenteket fizikai sorrendben olvassa és nem akadályozza a futó tranzakciókat, mivel nem zárolja a mentett adatokat. Továbbá a mentés befejezésének pillanatában érvényes állapotot tükröz: azok a tranzak-

ciók, amelyek a mentés befejezése előtt érvényesítődtek, a mentés visszatöltésekor megmaradnak, míg a lezáratlan tranzakciók visszagörgetődnek.

Megjegyzés: A fentiek szerint a fuzzy backup elmenti a tranzakciónapló egy részét is. Azonban ez nem tekinthető logmentésnek és nem üríti a tranzakciónaplót! A tranzakciónapló mentése és ürítése a BACKUP LOG paranccsal végezhető.

Az SQL Server 2000 adatbázisok mentéséről *(is)* sok hasznos információt tartalmaz a Microsoft® SQL Server™ 2000 High Availability című MSPRESS könyv. A könyv 9. fejezete, amely a mentés helyreállítás alapjaival foglalkozik, letölthető a **[sqlskills]** címről.

Fájl és fájlcsoport mentések

Nagyobb adatbázisok esetén előfordulhat, hogy a teljes adatbázis mentésének ideje túlságosan hosszúra nyúlik. Első pillantásra ez nem tűnik különösebben nagy problémának, mivel az adatbázis mentése nem blokkolja a futó tranzakciókat, és megfelelően gyors diszkek esetén észrevehető teljesítménycsökkenést sem okoz. Azonban vannak olyan tevékenységek, amelyek nem történhetnek az adatbázis mentése közben:

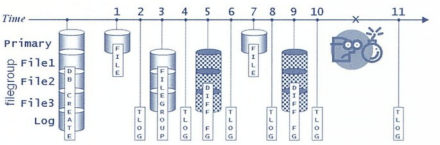
- Nem tudjuk menteni *(és üríteni)* a tranzakciónaplót, hiszen a fuzzy backup a tranzakciónaplóra is épít. Ennek következtében ha egy több óráig futó mentés közben történik lemezmeghibásodás, csak a régebbi mentésekre hagyatkozhatunk. Ettől még lehetséges az utolsó pillanatig érvényes helyreállítás, de minél régebbi mentésből indulunk ki, annál tovább tart.
- Nem tudjuk változtatni az adatbázis méretét. Ha megtelnek az adatfájlok, vagy a logfájl, sem az autogrow, sem a manuális ALTER DATABASE nem segít. *(Csökkenteni sem tudjuk az adatfájlok méretét, de az kevéssé zavaró.)*

Előrelátó tervezéssel ezek a korlátok nem okoznak gondot, de 500 GB fölött már többnyire egyéb megoldásokra, más mentési stratégiára van szükség.

Az egyik lehetséges megoldás a split mirror *(megtört tükrözésen alapuló)* mentés. Egyes hardvergyártók az SQL Server Virtual Device Interface *(VDI)* for Backup specifikációra alapozva olyan megoldásokat készítenek, amellyel a több terabájtos adatbázisok mentése is csak néhány másodpercet vesz igénybe. Tulajdonképpen nem mentés történik, hanem a többszörösen tükrözött diszkek egyik példányát leválasztják az SQL Serverről. A leválasztás idejére az SQL Server szüneteltetni a diszkre írást, hogy a félig írt lapokat *(torn page-eket)* elkerüljék. Ezután a leválasztott adatbázist egy másik géphez kapcsolhatjuk, szalagra menthetjük tetszés szerint. Elegáns, de költséges megoldás.

Olcsóbb, de több szervezést igénylő megoldás az egyes fájlok, vagy fájlcsoportok mentése. Az idei, barcelonai TechEd konferencián a DAT335 sorszámú előadás tartalmazta a következő példát.

A példához tartozó SQL szkript letölthető az előadó, Kimberly L. Tripp honlapjáról **[sqlskills]**. Haladhatunk az → enter events → Past Events → Microsoft TechEd Barcelona → SQL Server™ 2000 Tips and Tricks for DBAs and Developers *(DAT335)* → demo scripts útvonalon, illetve használhatjuk a **[sqlfilesave]** URL-t.



A fenti adatbázis egy .mdf, három .ldf és egy .log fájlból áll. Az első mentések sorozata tartalmaz fájlmentést *(a PRIMARY csoportban található .mdf fájlról)*, tranzakciónapló mentést, fájlcsoport és fájlcsoport differenciális mentéseket. Észrevehetjük, hogy teljes adatbázismentés soha nem készül, ennek ellenére a hiba pillanatáig érvényes helyreállítást tudunk végezni.

A helyreállítás menete a következő:

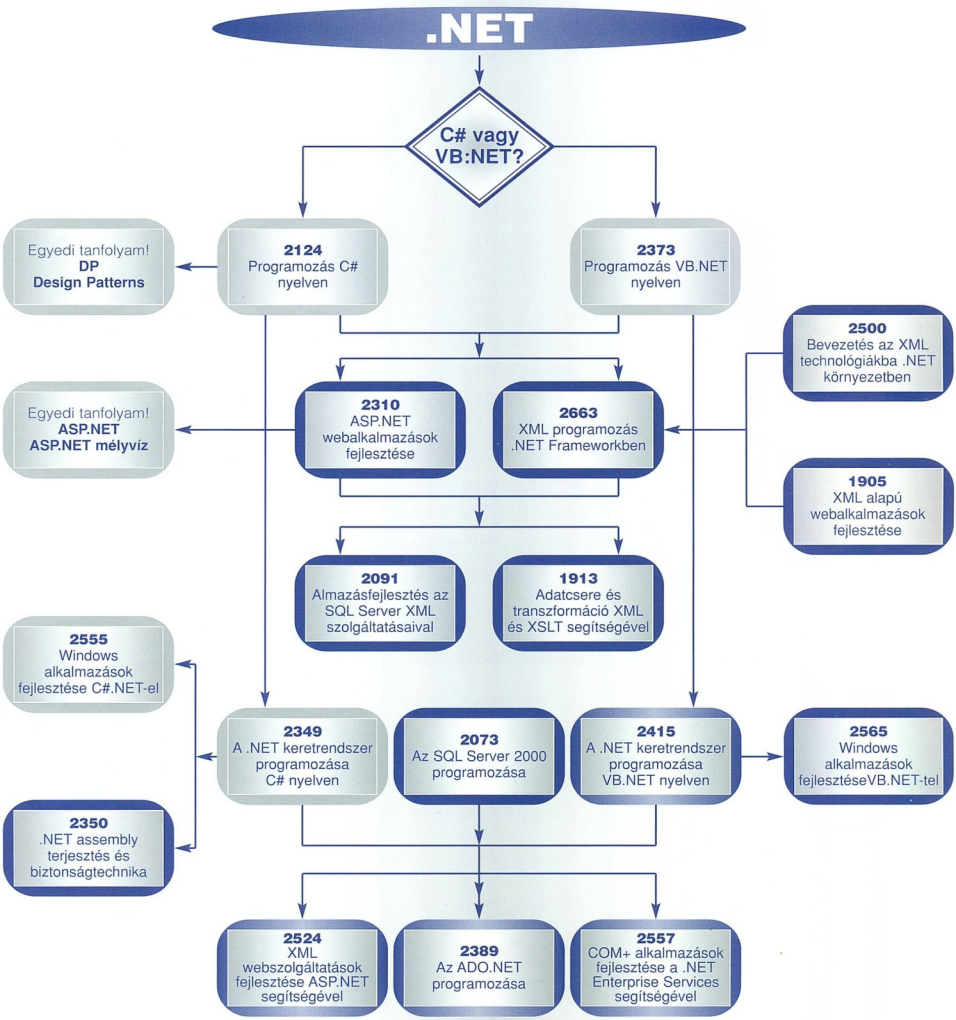
- a tranzakciónapló végét – ez a *(11)* sorszámú mentés.
- Helyreállítjuk az adatbázis szerkezetét. Ehhez betöltjük az elsődleges adatbázis fájl utolsó mentését *(7)* és a fájlcsoport utolsó teljes mentését *(3)*.
- A gyorsabb helyreállítás érdekében betöltjük a legfrissebb differenciális mentést *(9)*.
- Betöltjük azokat a logmentéseket, amelyek az adatbázis konzisztens állapotba hozásához szükségesek. Az ábra alapján valószínűleg a *(8)*, *(10)* és *(11)* sorszámú mentésekre lesz szükség, de elképzelhető, hogy korábbi log mentésekre is szükségünk lesz, ha az adatbázison hosszú tranzakciók futottak.

Hogyan állapíthatjuk meg, meddig kell visszamenni a logmentések sorában?
A RESTORE HEADERONLY SQL parancs kilistázza a mentés*(ek)* jellemzőit, köztük a FirstLSN-t. Fájl, fájlcsoport és adatbázis mentés esetén a FirstLSN az adott mentés végrehajtása alatt nyitott állapotú legöregebb tranzakció LSN-je. Log mentés esetén a FirstLSN annak a legrégebbi tranzakciónak az LSN-je, amelyet az adott log mentés teljes egészében tartalmaz. Azzal a logmentéssel kell kezdenünk az adatbázis konzisztenssé tételét, amelynek MinLSN-je kisebb, mint korábban betöltött fájl mentés *(7)* MinLSN-je és kisebb, mint a betöltött legfrissebb fájlcsoport differenciális mentés *(9)* MinLSN-je.

Bonyolultnak tűnik? Szerencsére, a RESTORE parancs figyelmeztet, ha nem elég korai logmentést akarunk visszatölteni. Mindenesetre, érdemes letölteni a szkripteket és kipróbálni.

Kószó Károly
rendszermérnök
karolyk@microsoft.com

Fejlesztői tanfolyamok térképe



www.netacademia.net • 1062 Budapest, Andrássy út 62. • Telefon: (1) 472-1214 • Fax: (1) 472-1215

Legyen Ön is a nagyok között!

Microsoft CERTIFIED Professional

Kedvezményes ősztől!

hivatalos Microsoft mérnök képzések

MCSE (rendszermérnök) képzés
az öt kötelező vizsgához
430.000 helyett már nettó **399.000 Ft-tól!!!**

Nagy, összetett informatikai rendszereket és hálózatokat üzemeltető szakemberek képzése, akiknek feladatuk lesz Windows 2000 alapú hálózatok teljeskörű adminisztrálása és támogatása, informatikai infrastruktúrák tervezése.

Október 27-től!

MCSA (adminisztrátor) képzés
a kötelező vizsgákhoz
270.000 Ft helyett már nettó **229.000 Ft-tól!**

A kisebb informatikai rendszereket és hálózatokat üzemeltető szakemberek számára ajánljuk, akiknek feladatuk lesz Windows 2000 alapú hálózatok telepítése, konfigurálása, adminisztrálása, karbantartása.

Október 27-től!

A képzéseken a Windows Server 2003-as rendszerek újdonságairól is szó esik.
Igény esetén a sorozat után kedvezményes Windows Server 2003 átképzést is igénybe vehet!

Microsoft .NET fejlesztői képzések
párhuzamosan a C# és Visual Basic .NET programnyelveken

Kezdő és haladó fejlesztők, programozók számára ajánlott képzések, akiknek feladatuk lesz összetett Windows alapú és webes alkalmazások készítése, fejlesztése. A tanfolyamok a kedvezményes Microsoft .NET fejlesztői képzéssorozat keretében is elérhetőek.

C#/VB.NET kezdő: 2003. október 13–17. SQL+ADO.NET: 2003. november 24–28.
Web Forms: 2003. december 1–5. Windows Forms: 2004. január 22–26.

A tanfolyamok biztosan indulnak. Csatlakozzon be Ön is!
Minél több tanfolyamra jelentkezik, annál olcsóbban juthat hozzá!

Válassza ki az Önnek megfelelő minősítést és képzési konstrukciót!

Hivatalos Microsoft tanfolyamokra épülő, rugalmas beosztású, kedvezményes konstrukciójú és árú képzéssorozatok. Különböző segédanyagokat, vizsgafelkészítő anyagokat tartalmazó oktatási konstrukciók és csomagok. Kedvezményes lehetőségek a szabadon választható vizsgára felkészítő tanfolyamokra.

A megadott árak a 25%-os ÁFÁ-t nem tartalmazzák.

Microsoft GOLD CERTIFIED Partner

Microsoft CERTIFIED Technical Education Center

SZÁMALK TOVÁBBKÉPZÉS

SZÁMALK

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

NetACADEMIA

A LEGJOBBAKAT TANÍTJUK.

Nem gyorsított!

Alapos .NET tudás + MCAD képesítés

Microsoft .NET fejlesztői tanfolyamcsomag több mint 30% kedvezménnyel a NetAcademiánál!

Az oktatási szemléletünkről

Gondolom Önnek nem kell bemutatnunk miért érdemes új fejlesztések esetén a .NET technológiákra alapozni. Inkább bemutatjuk mi mit nyújtunk Önnek a tanulásban, mivel különböztetjük meg magunkat versenytársainktól.

Mi nem kínálunk gyorsított tanfolyamokat!

Több mint **1000 óra .NET oktatási** és **több száz óra szaktanácsadási tapasztalatunk** alapján rengeteg kiegészítő, plusz információval látjuk el a hallgatókat, amelyeket lehetetlen gyorsított tanfolyamon átadni (még a normál ütemezéssel is nehéz). Emellett tudjuk, hogy időre van szükség a rengeteg új információ *valódi* megértéséhez, ezért a tanfolyamokat úgy szerveztük, hogy lehetőleg mindig legyen egy hét pihenő két témakör között.

Mi a megértésre és a magas színvonalra helyezzük a hangsúlyt, nem a sebességre!

A tanuláshoz részletes stratégiát dolgoztuk ki, amely a http://www.netacademia.net/training/tandotnet.aspx címen olvasható.

És mi van a kedvezményekkel?

Mint minden nagyobb volumenű megrendelésnél ezen csomagjainkra is jelentős kedvezményt adunk. Az alábbi táblázatban látható, hogy mindkét tanfolyamcsomag megrendelése esetén legalább 30% kedvezményt adunk.

Milyen vizsgákat foglal magában az MCAD és MCSD minősítés?

Az alkalmazásfejlesztői címhez (MCAD) két kötelező és egy szabadon választott vizsgát kell letenni. A kötelező vizsgák Windows *vagy* Webalkalmazás fejlesztési ismereteket (sötétebb sorok) és Webszerviz technológiákat kérnek számon C# vagy VB.NET nyelven. A választható vizsgák közül a magyarországi viszonyok között is **jól hasznosítható** SQL Server 2000 programozási vizsgát javasoljuk. Az MCSD fejlesztőmérnöki címhez Web *és* Windows alkalmazásfejlesztési ismeretekre is szükség van, és emellett egy tervezési vizsgát is le kell tenni, plusz egy választható vizsga az előbbihez hasonlóan.

A tanfolyamok oktatója **Soczó Zsolt**** MCSE, MCSD.NET, MCAD, MCDBA, MCT. Időpontok, részletes tematika és teszt vizsgakérdések online kiértékeléssel a http://www.netacademia.net/training címen találhatók.

A következő táblázatban összegyűjtöttük az általunk javasolt képzési utakat **C#** vagy **VB.NET** nyelvi alapokon.

Témakör	Tanfolyam C# (VB.NET)	Hossz [nap]	Kapcsolódó vizsga C# (VB.NET)	Listaár [Ft]	Következő időpont
Webalk. fejlesztése	2310 (2310)	5	70-315, 70-320 (70-305, 70-310)	160,000	2003. 11. 03. 2003. 02. 02.
Windows alk. fejlesztés	2555 (2565)	5	70-316 (70-306)	160,000	2003. 11. 24. 2003. 03. 22.
ADO.NET	2389 (2389)	3	Valamennyi	135,000	2003. 11. 17. 2003. 02. 23.
WebServices	2524 (2524)	3	70-315, 70-320 (70-305, 70-310)	135,000	2003. 12. 01. 2003. 03. 01.
.NET Framework	2349 (2415)	5	Valamennyi	160,000	2004. 02. 09. 2003. 02. 09.
MCAD csomag	**4 tanfolyam**	**Σ16**		**400,000** 590,000 helyett	
MCSD csomag	**5 tanfolyam**	**Σ21**		**500,000** 750,000 helyett	

Áraink csak 4 illetve 5 tanfolyam együttes megrendelése esetén érvényesek!

Minden tanfolyamárat 25% ÁFA terhel.

Jelentkezést faxon vagy levélben fogadunk el az alábbi címen/faxszámon:

H-1062 Budapest, Andrássy út 62. • Tel.: + 36 1 472 1214 Fax: +36 1 472 1215 • E-mail: info@netacademia.net • http://www.netacademia.net