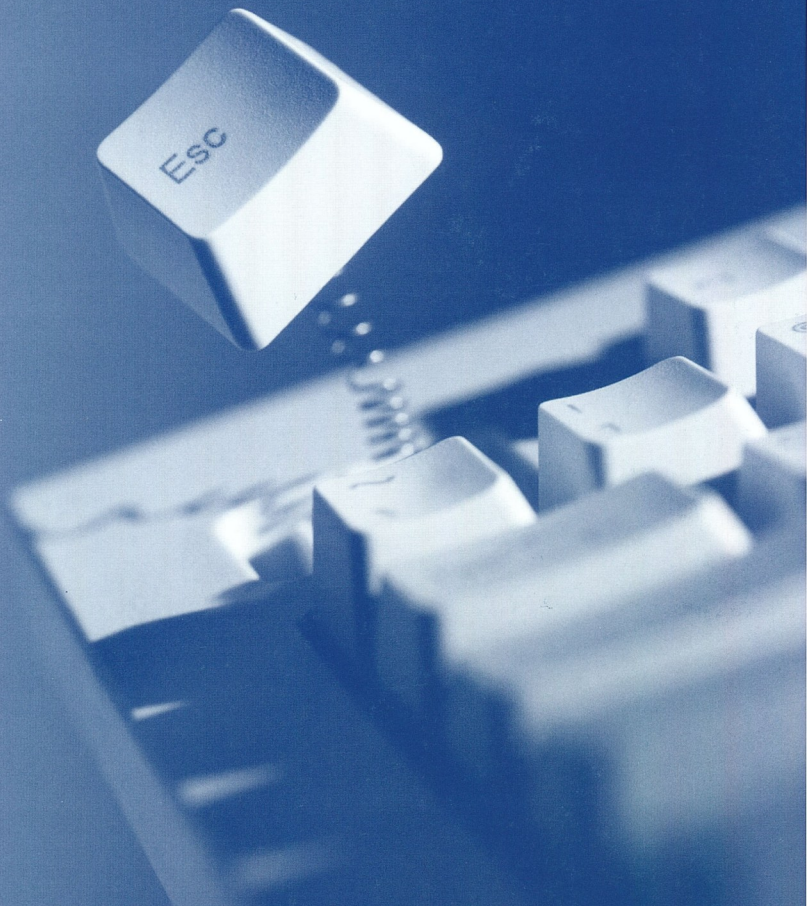
tech.net

working with windows



IV. / 09. szám
2003. szeptember
1344 Ft

ISSN 15865185 09
9 771586 518005

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

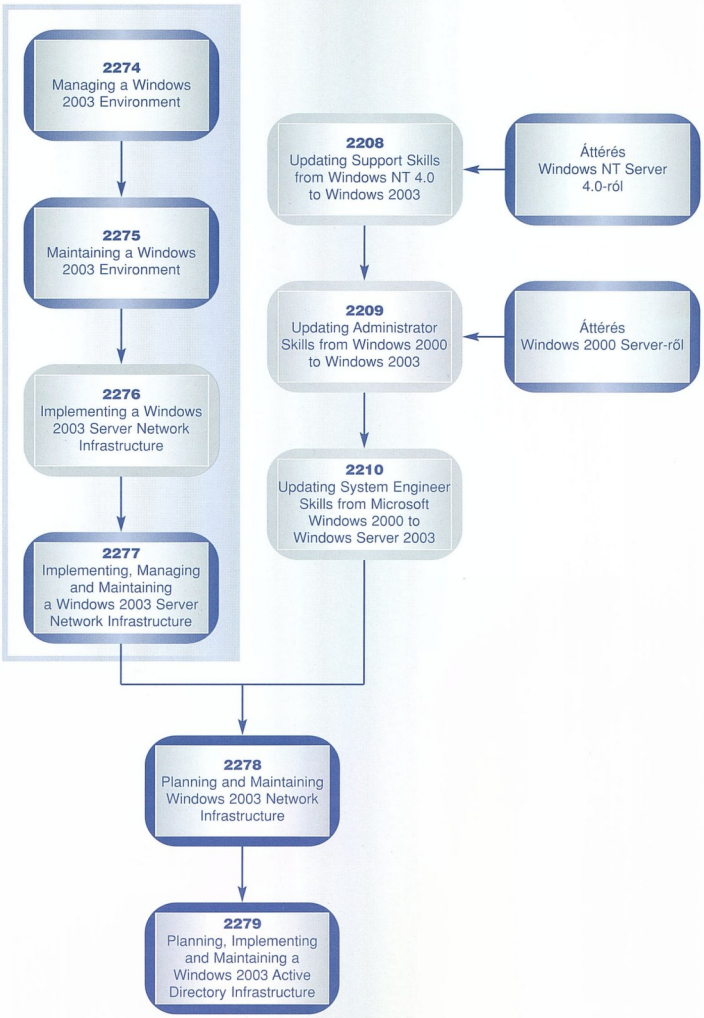
Hirdetésfelvétel: **Szívós Éva**
Telefon: 472-1214
Fax: 472-1215
info@netacademia.net

Nyomdai előkészítés:
Ars Luna Bt.
Vezető: Dobák Ildikó

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

ISSN 1586-5185

Windows 2003 Server rendszergazdai tanfolyamok térképe



Az újdonság varázsa

A Windows 2003 rejtett meglepetései

Akit hidegenhagynak a szerkeszthető X.509 sablonok, a WMI-filterek, az Active Directory Application Partition és a többi „marhaság", az ne tanuljon. De készüljön fel néhány váratlan kudarcélményre, mert ezeken kívül egy-két bit még átbillent, mire 2000-ből 2003-ba jutottunk. A legegyszerűbb feladatok is okozhatnak meglepetéseket!

Akár egy könyvtár megosztása is megoldhatatlan rejtély elé állíthat bennünket, ha *(vakon)* sikerül eltalálnunk egy olyan pontot az operációs rendszerben, ami már nem úgy működik. Egyszerű feladat: a munkaállomásokon futó víruskeresők számára kell készítenünk az egyik Windows 2003 kiszolgálón egy megosztott könyvtárat, hogy onnan töltsék le a frissítéseiket.

Fogjuk a könyvtárat, jobbklikk, megosztás, beírjuk a megosztás nevét és OK. Ennek mennie kell, mivel a megosztásokon a bántatlan, alapértelmezett jogosultság – mint tudjuk – Everyone -> Full Control.

Rosszul tudjuk. A biztonságos üzemeltetés érdekében ugyanis pontosan 2003 február 17 óta az alapértelmezett megosztásjog: Everyone -> Read!

Ezt a „hibát" viszonylag könnyen orvosolhatjuk: jobbklikk, Permissions, és engedélyezzük a Change *(változtatás)* jogot is. A víruskereséink azonban csak nem férnek hozzá a frissítési adatbázisaikhoz. Most mit tegyünk? Ez már fogósabb probléma, tekintettel arra, hogy semelyik bátortalan kísérletünk sem sikerül. Nem elég neki a Full Control sem. Itt bizony valamilyen kompatibilitási galibába futottunk!

Szakember legyen a talpán, aki ezt a problémát anélkül képes megoldani, hogy tudná: alapvetően megváltozott az Everyone csoport! 2003 előtt igaz volt az alábbi képlet:

```
Everyone = Authenticated Users + Anonymous
```

A redmondi dizájnerek úgy gondolták, hogy aki névtelen kapcsolatot nem akar engedélyezni, majd ügyesen áttér az NT4 SP3-ban bevezetett Authenticated Users csoport használatára. A felmérések tanúsága szerint ezt a lépést a rendszergazdák 0 *(nulla)* százaléka tette meg. Mivel azonban Bill úgy döntött, a Windows lesz a világ legbiztonságosabb operációs rendszere, a lusta rendszergazdák háta mögött cselt forralt, és kivette az Everyone csoportból a névteleneket. Tehát ma így fest az egyenlet:

```
Everyone = Authenticated Users
```

Ez a „hiba" oka. A SYSTEM fiók alatt futó víruskeresők névtelen felhasználónak számítanak. Ennek elhárítása azonban vagy nem egyszerű *(NTLM-világ)*, vagy nem biztonságos *(GPO)*, vagy nem ismert *(Kerberos)*. Lássuk az eseteket sorjában.

- A nem egyszerű módszerre NT4 alatt van szükség, mert ha meg szeretnénk szüntetni a víruskereső anonimitását, bizony minden gépen át kell állítanunk a szolgáltatást, hogy ne SYSTEM, hanem valamilyen valós felhasználó nevében fusson.
- A nem biztonságos megoldás nem más, mint az Everyone csoport visszakapcsolása 2000-kompatibilis üzemmódba. Ehhez tartományi csoportházirendet használhatunk, így egy laza mozdulattal akár az összes Windows 2003-on rést üthetünk a pajzson. Anonimusz jöhet.
- A nem ismert módszer ott használható, ahol a munkaállomások Kerberos autentikációt használnak. Ebben az esetben beállíthatjuk, hogy a névtelennek számító SYSTEM fiók helyett a munkaállomás szolgáltatásai a számítógépfiókot *(Computer Account)* használják önazonosításra *(például egy MASINA nevű gép esetén a bejelentkezési név MASINA$ lesz)*. Ezt a címtárban lévő számítógépen állíthatjuk be. Jobbklikk, tulajdonságok, Trust this computer for delegation.

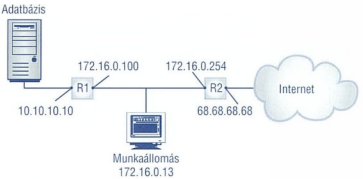
Ezt a rövid kis példát ízelítőnek szántam. Miközben a Windows 2003-ban bolyongtam, rengeteg ilyen apróságra bukkantam. Például:

- A GPO-ban átnevezték a Log On Locally jogot Allow Log On Locallyra, hogy nehogy azonnal megtaláljuk. Cserében viszont kizárták annak lehetőségét, hogy kizárjuk magunkat a tartományból: ha másnak nem is, az Administratorsnak mindenképpen bent kell lennie a listában, enélkül nem lehet leokézni.
- A DHCP-kiszolgálón suttyomban megjelent egy új opció, a 249-es. Ennek hasznlatáról lásd a negyedik oldalon kezdődő cikkemet.
- Az RRAS-ban már nem csak registry hekkeléssel lehet rávenni az L2TP/IPSec párost, hogy felejtse el a tanúsítványalapú azonosítást, hanem térjen át preshared keyre *(közös titok)*

A többit majd szóban, a tanfolyamokon.

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

2003. 09. szám

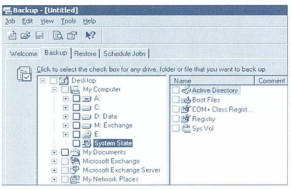


Komámasszony, hol az ajtó?

Enyhén összetett hálózat útválasztásának többféle megoldása

Talán nem ritka az a hálózatkialakítási forma, amikor a munkaállomásoknak egyszerre kétfelé kell „elhagyniuk" a tett színhelyét, azaz két hálózati kijárat *(gateway)* használata között kell dönteniük. Vajon elég intelligensek ehhez, vagy *(rossz kijárat használata miatt)* mindenképpen duplázott hálózati forgalommal kell számolnunk?

4. oldal



SBS 2000 helyreállítása

Mi történik, ha hiba esetén egyedül van egy AD controller?

Rémálom a szerverszobában: elszállt az Active Directory, ráadásul mindez egy SBS 2000 szerveren... Mentés, visszaállítás és minden ami mögötte van!

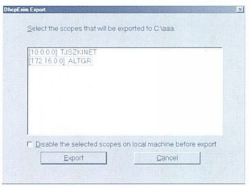
6. oldal

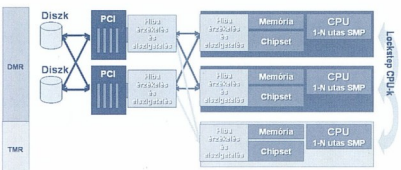
Gyógyszeresdoboz IV.

Windows 2000 Server Resource Kit

A Network Management Tools II., azaz a sorozat előző részében taglalt, felhasználókkal, csoportokkal és megosztásokkal kapcsolatos eszközök átvilágítása után, most tekintsünk bele egy másik világba.

9. oldal





Megállás nélkül

99,999%-os rendelkezésreállás Windows operációs rendszerrel?

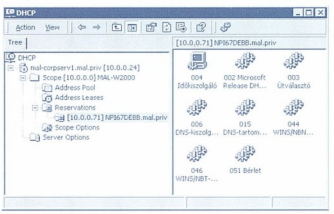
A cél az „ötkilences" rendelkezésre-állás. Évente 5 perc leállás. Ilyenkor aztán beindul az agyalás: Hogyan? Farkasokkal táncolunk, clustert építünk és fürtözünk? Ide ez már nem elég. Ez már egy más kategória, itt a fürt hamar elvérzik. Ide Stratus kell!

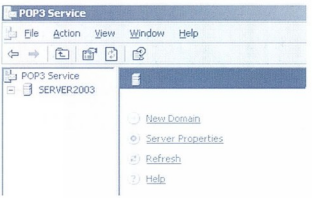
11. oldal

A DHCP rejtett szépségei – IV.

Egy végső lendülettel megbirkózunk a DHCP utolsó nagy fogalomkörével, vagyis az osztályokkal és opciókkal. A cikksorozat második részében már megemlítettük a legfontosabbakat, de túlságosan nem merültünk el a részletekben. Most viszont épp ezt fogjuk tenni, hogy megértsük, milyen eszközeink vannak az ügyfelek finomhangolására, s egyáltalán: mi mindenre képes a DHCP...

13. oldal





Internet Information Services 6.0, IV. rész

Levelező-szolgáltatások a Windows Server 2003-ban: a POP3

Régóta vártunk arra, hogy a levelek fogadásáért és kiküldéséért felelős SMTP-szolgáltatás mellett a felhasználók postafiókját és az ahhoz való hozzáférést biztosító POP3 is megjelenjen a palettán. A Windows Server 2003-ban végre itt van – kicsit butuska, kicsit egyszerű, de a mienk :-)

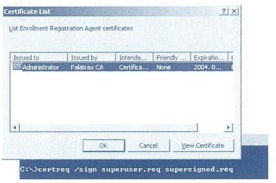
16. oldal

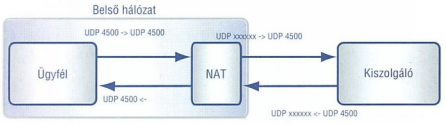
Tanúsítványkiadók a Windowsban

Mi van egy tanúsítványban?

Cikkünk következő részében először az X.509 digitális tanúsítványok mélyreható boncolgatására kerül sor, majd visszatérünk a Windows Server 2003, Enterprise Edition tanúsítványkiadójának újdonságaira.

22. oldal



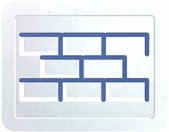


NAT-T: IPSec NAT-on keresztül

IPSec NAT Traversal – az IPSec új üzemmódja

A Windows 2000-ben bevezetett IPSec protokoll egyik nagy hátránya volt, hogy az általa titkosított csatorna nem volt képes áthaladni hálózati címfordító *(NAT – Network Address Translation)* eszközökön. A Windows 2003 már tartalmazza, Windows 2000-hez és XP-hez pedig letölthető azt a bővítés, ami megoldja ezt a problémát is.

27. oldal



Biztonságos aláíráskezelő alkalmazás (BALE) készítése IV.

Adatok és komponensek

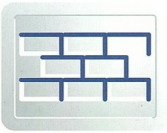
A hosszúra nyúlt bevezetés után végre eljutottunk a gyakorló informatikus kollégák számára minden bizonnyal kedvesebb témákhoz. Jelen részben az aláíráskezelő alkalmazás által használt adatokról, az aláíráslétrehozó alkalmazás ezeket hasznosító komponenseiről, valamint az alkalmazással szemben támasztott követelményekről lesz szó. Egyben folytatódik a mindennek nevet adunkszakkör, ami most ér igazi csúcspontjára, hiszen most egyes fogalmaknak új elnevezést adunk. Tessék kapaszkodni, mert aki eddig netán értette, az most könnyen elveszítheti a fonalat.

30. oldal

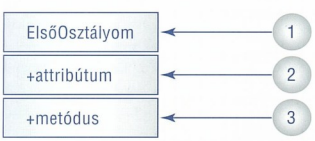
Elliptikus görbe kriptorendszerek II. rész

Elliptic Curve Cryptography – algoritmusok és a gyakorlat

Egy jó ideje egyre több helyen találkozhatunk az ECC betűhármassal, mint egy kriptorendszer jelölésével. A cikk első részében áttekintettük e viszonylag fiatal kriptorendszer elvi alapjait, most jöjjön néhány gyakorlati példa és algoritmus!



36. oldal



Objektumorientált alkalmazásfejlesztés

Az UML

Sok szó esett már az UML-ről, és talán még mindig vannak, akik számára nem egyértelmű, mit is takar ez a három betű tulajdonképpen. Számukra szeretnék egy kis segítséget nyújtani, hogy tisztább legyen a kép, és ne rémüljenek meg, ha egy osztálydiagram feltűnik valahol.

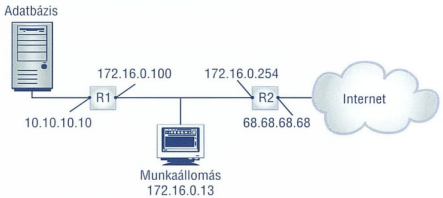
40. oldal

Komámasszony, hol az ajtó?

Enyhén összetett hálózat útválasztásának többféle megoldása

Talán nem ritka az a hálózatkialakítási forma, amikor a munkaállomásoknak egyszerre kétfelé kell „elhagyniuk" a tett színhelyét, azaz két hálózati kijárat *(gateway)* használata között kell dönteniük. Vajon elég intelligensek ehhez, vagy *(rossz kijárat használata miatt)* mindenképpen duplázott hálózati forgalommal kell számolnunk?

Elsőként vázolnám a problémát, vagyis felrajzolom azt a hálózatot, amelyben a munkaállomásnak el kell döntenie, vajon a rendelkezésre álló routerek közül adott esetben melyiket válassza?

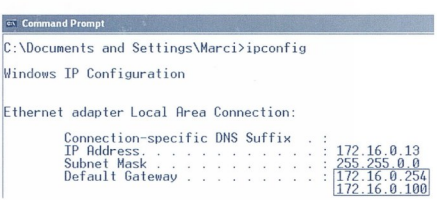


Egy tipikus, adatközponttal és Internetkapcsolattal rendelkező hálózat felépítése

Az ábra egyszerűsége miatt talán magyarázatra szorul, hogy hol itt a gond? Mivel a munkaállomás gazdája bizonyosan szeretne Internetezni, beállítjuk az R2-vel jelölt útválasztót, mint alapértelmezett átjárót *(Default Gateway)*. Az adatközpontban található adatbázis eléréséhez pedig R1-et is beállítjuk alapértelmezett átjáróként. A TCP/IP tulajdonságok, Advanced fülén lehet felvenni a sokadik átjárót:

Egynél több alapértelmezett átjáró. Bizarr ötlet?

Egy sima IPCONFIG parancs kiadásával le is ellenőrizhetjük, hogy mit műveltünk:

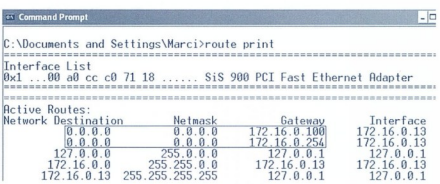


A két átjáró azt igazolja, hogy a dolog működik... De!

Ha most a munkaállomásról hálózati forgalmat indítunk mindkét kritikus irányba, Network Monitorral láthatjuk, hogy a gép nem veszi figyelembe a második átjárót. Nem nyert!

Dead Gateway Detection

Első körön sikerült belefutnunk a TCP/IP implementáció egyik gyakran félreértett tulajdonságába. Nevezetesen abba, hogy ha egynél több alapértelmezett átjárót adunk meg, akkor a Windows az elérendő cél függvényében majd okosan hol egyiket, hol másikat fogja használni. Hogy ez biztosan nem így van, sőt el sem képzelhető, hogy így működjön, az úgynevezett útvonaltábla *(routing table)* szolgál bizonyítékkal. Nézzük meg *(ROUTE PRINT)*, mit is tartalmaz egy ilyen duplaátjárós gép úttáblája!



A két default gateway megjelenése az útvonaltáblában

A „hiba" oka világosan leolvasható: a Default Gateway csupanullás bejegyzései azt jelentik, hogy minden olyan csomagot, amiről nem tudjuk halál pontosan eldönteni, hogy hova menjen, dobjuk csak neki.

Csakhogy mindkét DG ugyanígy került be a táblába, egyik sorban sincs egy fikarcnyi „intelligencia" sem, így nem lehet eldönteni, hogy melyik vezet az Internetre, és melyik az adatközpontba!

Az ábráról egyébként lemaradt mégegy paraméter: a metric. Ezt azt határozza meg, hogy az egyes útvonalakat milyen prioritás szerint használja a munkaállomás. Nos, ez sem visz előre, mivel mindkét DG-nél a metric=30. Elvileg így egyenlő eséllyel kellene használnia mindkettőt *(terhelésmegosztás, load balance)*, de mégcsak nem is ez történik. Hanem a következő: amíg az első DG él és virul, a másodikat nem veszi használatba. Ennek a működésnek a hivatalos neve: Dead Gateway Detection. Nem mellesleg: ha egynél több DNS-kiszolgálót állítunk be, azok közül is mindaddig a legelsőt használja, amíg az meg nem döglik.

Jelenleg tehát mivel a 172.16.0.254-es átjáró van elöl, a munkaállomás minden kimenő csomagot R2-nek küld, amit az

szétválogat, és az adatközpontba tartó csomagokat egyesével átküldi R1-nek. Ezt szaknyelven úgy nevezik: duplázott hálózati forgalom!

Megoldás I. - az útvonaltábla kézi módosítása

Ha sikerülne felokosítanunk az útvonaltáblát, és kiderülne, hogy pontosan mely címekre menő csomagok érdekében érdemes használni egyik vagy másik routert, sokkal előrébb lennénk. Mivel az összes Internetes cím megadása gyakorlatilag lehetetlen, jobban járnánk, ha az adatközpont címeire tanítanánk meg a kis buta munkaállomást. Ehhez először is vegyük ki a TCP/IP-beállítások közül a második átjárót, majd a következő bűvös paranccsal adjuk meg, hogy minden 10-es címre menő csomagot R1-re kell továbbítani:

```
route add 10.0.0.0 mask 255.0.0.0 172.16.0.100
```

Ezután az útválasztótábla így néz ki *(csak a lényeget emeltem ki)*:

```
Network Destination        Netmask          Gateway       Interface
          0.0.0.0          0.0.0.0    172.16.0.254    172.16.0.13
         10.0.0.0        255.0.0.0    172.16.0.100    172.16.0.13
```

Egy konkrét hálózat (10.0.0.0) szerepel az útvonaltáblában

És ezzel már jól fog működni a munkaállomás: az adatközpontba menő csomagokat R1-nek, az Internetre menőket R2 felé továbbítja.

Nincs is más hátra, mint végigmenni a hálózaton, és mind az 534 munkaállomáson kiadni a route add parancsot. Természetesen /P kapcsolóval, hogy a hatás maradandó legyen *(persistent route)*.

Nem lehetne ezt másképp? Központosítva? Automatizálva? De, lehetne, mégpedig többféle módon is.

Megoldás II. – az útvonaltábla módosítása DHCP-vel

A cikkben vázolt útválasztási probléma mintegy hat éve szúrja a szememet. Hat évvel ezelőtt még nem volt rá gusztusos, automatikus megoldás, mert sem a most említendő DHCP, sem a III. megoldás nem működött NT4-gyel. Gusztustalan persze volt: egy route add parancsot tartalmazó login script. Fuj.

Pedig már akkor kiszúrtam magamnak, hogy a DHCP opciók között virít egy 33-as sorszámú, Static Route nevű csoda, amit pont nekünk találtak ki. Mi más lehet a static route, mint egy „kézi" bejegyzés a DHCP-ügyfelek útvonaltáblájába?

A 33-as pont az, de több sebből vérzik. Az első seb abból fakad, hogy *(anno 1996)* a DHCP-ügyfél Windows nem veszi figyelembe, hiába állítjuk be. Csak és kizárólag azt a 6-7 opciót fogadja el, amit jelen számunk DHCP cikkében is olvashatnak.

De nagyot változott a világ a múlt század óta, és ma már ezt is elfogadják a Windowsok *(Windows 2000-től kezdődően)*. Viszont van egy második seb is: a 33-as opció CIDR előtti. *(Az 1993-as Classless Interdomain Routing szabvány az IP-cím pocsékolás megakadályozására engedélyezi, hogy egy IP-címtartományt tetszőleges subnet maskkal ripityára aprítsunk.)* A CIDR-előttiség azt jelenti, hogy a 33-as opcionál nem lehet subnet maskot megadni az útvonal megadásakor, hanem az IP-cím osztályából *(A, B, C)* következő maszkot rendeli hozzá a rendszer. Ez ma már nem számít korszerűnek, mondhatni hajítófát sem ér.

Csakhogy ma 2003-at írunk. A technológia fejlettsége trilliószorosa a három évvel ezelőttinek, ezért vettem a fáradságot, és belekukkantottam a Windows 2003 DHCP-kiszolgálójába, vajon történ-e ezen a téren változás? Igen, igen, igen!

Született egy új opció *(a 249-es)*, amit PONT nekünk találtak ki! Végre megoldható a munkaállomások útvonaltáblájának korrekt módosítása! Milyen kár, hogy erre ma már semmi szükség! Hogy miért? Mert a III. megoldás a nyerő.

Megoldás III. Öntanuló hálózat

Elgondolkodtak-e már azon, milyen jó is lenne egy olyan hálózat, amely felfedezi a saját problémáit, és segít magán? Ha itt még nem is tartunk, de a duplázott hálózati forgalom ellen van automatikus eljárás. Ez pedig a munkaállomások „távtanítása" a routerek révén.

Dobd ki azt a routert, ami nem észleli a duplázott hálózati forgalmat, és nem szól vissza a feladónak, hogy „hé öcsi, rossz ajtón kopogtatsz!".

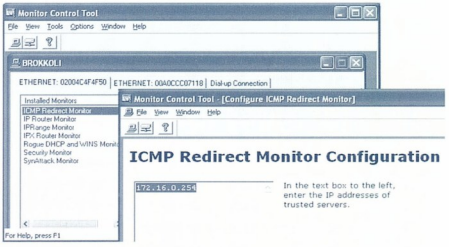
Persze ettől még nem javul meg egy csapásra minden, mert a munkaállomás nem köteles figyelembe venni a baráti figyelmeztetést, bár a Windowsok NT4 SP3 óta figyelembe veszik, és hálásan át is vezetik útvonaltáblájukat – pont úgy, ahogy az imént ezt puszta kézzel tettük.

Mi is ez a csodaüzenet, ami ilyen fantasztikus hatást vált ki a gépekből? Nem fogják elhinni: ICMP *(Internet Control Message Protocol)*. Ez ugyanaz a protokoll, mint amit a PING használ, csak annak egy másik ága, az ICMP Redirect.

A Redirect „parancs" segítségével az erre felkészített routerek utasíthatják klientúrájukat, hogy melyik csomagot merre dobálják. A Windowsok pedig engedelmeskednek. Talán túlságosan is engedelmesek. Mindegy nekik, ki küldte a Redirect parancsot, ők átállítják útvonaltáblájukat. Egy-két jólirányzott ICPM Redirect üzenettel pillanatok alatt le lehet szakítani egy-egy alhálózatot a hálózat többi részéről.

Ennek elkerülésére jelenleg egyetlen módszer ismeretes: a Network Monitor teljes verziójához adott Monitor Control Tool. Ez az eszköz különböző kellemetlen és nem várt hálózati jelenségek felbukkanását figyeli, és riadalmat kelt, ha gyanús eseményt észlel. *(Nem, nem küld semmit. Amikor ez készült, még nem volt divat az email-értesítés.)*

Egyébként bámulatosan egyszerű eszköz, csak kettőt kattintunk a kiválasztott ellenőrön, és ő a parancs megadása után azonnal szolgálatba lép:



ICMP Redirect monitor a Monitor Control Toolban

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

SBS 2000 helyreállítása

Mi történik, ha hiba esetén egyedül van egy AD controller?

Rémálom a szerverszobában: elszállt az Active Directory, ráadásul mindez egy SBS 2000 szerveren... Mentés, visszaállítás és minden ami mögötte van!

Az egész úgy kezdődött, hogy hétfő reggel olyan állapot fogadott munkahelyemen, ahogyan azt a legkevésbé vártam volna. Ahogy beléptem, megláttam, hogy a szerverszoba ajtaja nyitva van és az a valaki, aki kinyitotta nincs bent, hanem – feltehetőleg tesztelés céljából – elment megnézni, hogy jó lette... Mármint annak a cselekvéssorozatnak a hatása amit az előbb megpróbált az egyik szerveren véghezvinni. Nekem, mint a hálózat rendszergazdájának ilyenkor összeugrik a gyomrom, és a probléma végleges megoldásáig úgy is marad. Kisvártatva egyik kedves kollégám jött szembe azzal, hogy nem működik a weboldal, amit többek között az érintett szerver szolgáltat.

Ekkor rövid kattintgatás után kiderült a legrosszabb: a Small Business Server 2000 teljesen felmondta a szolgálatot. Sőt, a kezdeti legjobb megoldás – a restart – csak rontott a helyzeten, mert ezután még kétszer magától újraindult. Igaz harmadikra összeszedte magát, és a rajta levő Exchange adatbázisok is felmountolódtak, de nem volt tökéletes a működése. Ennek a következő jelei voltak: bármilyen Explorer indítása a teljes task lehalásához vezetett, a desktop-explorer minden kísérletre azonnal újraindult. Ám a legrosszabb az volt, hogy időnként a következő üzenet jelent meg egy kis figyelmeztető ablakban: „Memory could not be written" *(!)* Ekkor még vicces kedvünkben voltunk, így elkezdtünk találgatni, hogy vajon miféle ürtevékenység folytán alakult át a RAM memória ROM típusúvá... ahogyan azt a fenti üzenetből elsőre gondoltuk.

De a viccelődés hamar abbamaradt, amikor láttuk, hogy mekkora a baj. Technikailag a szerver egy igen erős vas *(DELL Power Edge 6400, dual Xeon, 2 GB RAM, 18 GB RAID1 HDD System és 70 GB RAID5 HDD Data)* a gyártó cég egyik kiemelkedő, csúcsminőségű modellje. De mégis, a hibaüzenetből arra gondoltunk, hogy hardverhibáról van szó, hiszen világosan megmondta, hogy nem tudja írni a memóriát. Mindezek mellett azt is vizsgáltam, hogy mi az, ami még nem működik. Elérhetetlen volt az IIS metabase, így nem szolgáltatta a weboldalakat, illetve képtelen voltam bármit csinálni a konzolon, mert minden fél percben elszállt az Explorer. Hamar kiderült, hogy ez így nem maradhat, hiszen minden tizedik másodpercben előállt a fentebb említett memóriaírási hibaüzenet is.

Úgy döntöttünk, hogy kihívjuk a hivatalos szervizt, kérve egy új memória modult, mivel még ekkor teljesen biztosak voltunk abban, hogy annak kicserélésével a probléma megoldódik majd. Ha így lett volna az én cikkem is csak eddig tartott volna, de nem így történt...

A másik modullal is ugyanígy viselkedett, és produkálta az auto-restartot is, ami szerver esetében igen-igen zavaró. Hiszen nem tudhattuk, hogy hány restart után indul el végre, és miért van, hogy néha nincs restart indulás előtt, néha meg kettő is van. Gondolom, nem vagyok egyedül azzal az érzéssel, hogy nem szeretem a számítástechnika világában az egyesek és nullák mellett olykor előforduló másfelet! De akármilyen hihetetlen is, ez a problémakör egyidős a szakmával...

Hát ez bizony nem hardverhiba!

Miután a memóriacsere nem segített, kitaláltuk, hogy a szerverben található ötféle BIOS és egyéb Firmware frissítése után egy tartalék HDD-re felteszünk egy szűz Windows 2000 szervert tesztelés céljából. Mondanom sem kell, hogy azzal semmi baja nem volt a memóriának, és minden tökéletesen működött. Itt kellett elfogadnunk azt a szomorú tényt, hogy a hiba szoftver-eredetű. Nem adtuk ám fel ilyen könnyen, mert még kipróbáltuk azt is, hogy a diszkeket egy pontosan ugyanilyen szerverbe betéve elindítottuk a rendszert. Sajnos ugyanúgy viselkedett, mint a másik vassal, azaz minden addig előforduló hibajelenség előjött itt is.

> Visszaállítani egy AD controllert ha egyedül volt és minden fontos adattal csak ő rendelkezett... Nem lehetetlen, de nem is egyszerű!

Ennél a pontnál kerültek tanácskozásaink előterébe a mentések. Egy évvel ezelőtt, amikor átadtam a szervert, a mentést úgy terveztem, hogy DAT kazettára *(napi cserélgetéssel)* egy héten öt mentés készül. Mivel a használt DAT mérete lehetővé tette, igencsak bőkezűen bántam a menteni valóval. Az alkalmazott szoftver a Veritas Backup Exec 8.6-os terméke, kiegészítve a hozzávaló Exchange 2000 Agenttel. Ezzel nagyon jól kézbentartott időzített mentéseket lehet készíteni és én szándékosan úgy állítottam be, hogy a lehető legtöbbet mentsen minden nap. Így redundáns információ is volt bőven, de megérte! Készült sima online adatbázisszintű mentés az Exchange-ről, illetve rögtön utána individuális mailboxonkénti mentés is. Az egész C *(System)* drive, illetve a System State is minden nap a kazettára került. Hogy ez miért fontos, arra most nem térek ki, mert az **[sbsreskit]** címen elérhető SBS 2000 Resource Kit Documentation részletesen foglalkozik a biztonságos mentés megtervezésével.

De egy hibát azért elkövettem, mégpedig azt, hogy nem volt a mentési tervben egy havi és mondjuk egy háromhavi mentési kazetta. Így minden hétfőn felülíródott a múlt hétfői, kedden az előző keddi anyag, és így tovább. Sokszor volt adatvisszaállítási kérés a felhasználóktól, és erre ez a módszer tö-

kéletesen megfelelt. De nem most! Ugyanis a hiba régebbi eredetű volt, mint egy hét, így a mentés összes adathordozója javában tartalmazta a bugot, mire észrevettem, hogy baj van. Az adatokkal, fájlokkal semmi gond nem volt, de az Active Directory csúnyán megsérült, így a System State backup is ezt az állapotot rögzítette a kazettára. Mivel a szerver egy SBS 2000 volt, a probléma is halmozottan jelentkezett.

Az Active Directory-tervezés és az SBS…

Ugye azt mindnyájan tudjuk, hogy nem jó – és a Microsoft által kifejezetten ellenjavallt – ha egy AD controller egyedül van. Továbbá az sem jó, ha a GC van egymagában és még ráadásul Exchange szerver is van rajta. Erre mindezek ismeretében kiadnak egy terméket, ami nem is működik máshogy! Ha akarjuk sem! Az SBS egyedül van, és úgy is marad. „Ha baj van, javítsd meg magad!”

Az egyetlen általa létrehozott domainben ő egyedül a kiskirály, és nem tűr meg maga mellett semmilyen más kiszolgálót sem. „Nagy baj lesz ha…” típusú ügyvédi, jogi szövegekkel teleírt ablakok tömkelegét szórta rám még attól is, amikor a hálózatban található másik domain DHCP szerverével került összetűzésbe. Ezek után persze csakis a mentésben bízhat a szegény adminisztrátor, nem úgy, mint egy „rendes” domain esetében, ahol legalább kettő tartományvezérlő tartja az AD adatbázist.

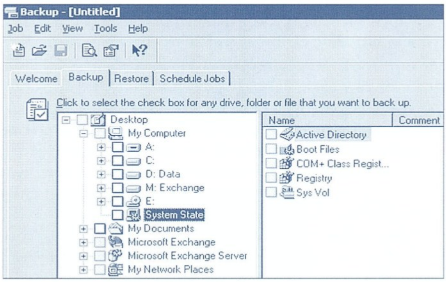
Persze mielőtt az ember elkezd egy teljes visszaállítást, megpróbál mindent, amit csak tehet. Mert elölről kezdeni egy évnyi munka után - brrr… ez szörnyen hangzik. Itt kezdődtek az álmatlan éjszakák.

Eredeti SBS W2K CD-ről indított Recovery-Install, System File Checker és társai. Persze közben egy 50 fős csapat – a felhasználói tábor – kényszeredett nyári szabadságon van, mert a mai informatikával telezsúfolt világunkban már szinte egy kávét sem főzünk meg, ha nem jó a szerver. Tudom, hogy az a sorstársam, aki látott már tétlen felhasználót az irodában téblábolni, mert nem tudja kinyitni az e-maileket, az tudja, hogy miről beszélek. Én meg persze ilyenkor megteszek mindent, hogy a lehető legrövidebb idő alatt elhárítsam a bajt: éjszakai próbálkozások, hogy esetleg reggelre már jó legyen, stb., stb. Nem először derítettem fényt a szervezetem végső határaira, ugyanis három nap alvás nélkül – és utána belázasodom. Ez van, ennyit tud földi porhüvelyem. A szerver persze soha nem alszik, nem fárad el, és addig nem hagy nyugodni, amíg meg nem javítom. Fontos megjegyezni, hogy ennek semmi de semmi értelme. Tiszta fejjel, kipihenten sokkal hatékonyabban lehet dolgozni, és kisebb az újabb hibák elkövetésének veszélye is. „De ha egyszer annyira jó lenne most gyorsan megoldani, és reggelre már működne is”… Ugye ismerős? Ördögi kör! Nem is ez vezetett sikerre. Mert persze az volt a végén, azért is írom ezt a cikket, mert elhatároztam, hogy ha sikerül végül helyrehozni ezt a szervert, megosztom a tech.net olvasóival az odáig vezető út minden örömét és bánatát.

A Directory Services Restore módban indított szerverre visszatettem az egyik legutolsó kazettáról a System State-et illetve a teljes rendszert és vártam, hogy megjavuljon. De ehelyett minden fontos service arról számolt be, hogy nem tud valamiért elindulni. Egyik ezért, a másik azért, de a lényeg, hogy szinte semmi sem ment. Ezután jött a repair a gyári CD-ről. Ekkor viszont az auto-restart vált folyamatossá mindig, bármelyik napi kazettával próbálkoztam is. *(Vegyük észre, hogy itt minden esetben egy teljes installról van szó, amit „elszúrtam” egy kazettáról visszahozott System State-tel, majd a Repairrel! Ennyi időt nem szabad ilyesmivel eltölteni, mert az ember az idegei épségével játszik közben!)* Egyszer kínomban megvártam, amíg egymás-után tízszer újraindult, hátha mégis megjön az esze közben. Örök tanulság: egy ilyen viselkedésű szerver akkor sem lehet jó, ha annak látszik. Ha el is indult volna, akkor sem hihettem volna róla mást, mint hogy véres a torka, és úgyis valami nagyon súlyos gondja van. Tudni kell eldönteni, hogy honnantól van reinstallról szó! *(Többnyire jóval hamarabb, mintsem ahogy azt bevalljuk magunknak.)*

Jöhet a reinstall!

Elfogadott tény, hogy az a tiszta, „tuti” megoldás, ha friss install után visszahozzuk az adatokat a mentésből, és megy minden tovább! Sajnos viszont a mentés nem volt teljesen használható – pontosan a legfontosabb, az Active Directory része nem.



Ha van jó System State backup, az sokat segíthet. Ha nincs, jöhet a manuális javítgatás…

Szerencsére, amikor ezer próbából egyszer nagy nehezen, de elindult a szerver, a felhasználók be tudtak lépni. Igaz, semmi mást nem tudtak csinálni, mert semmi más nem működött, de ez mégis azt mutatta, hogy a csoportok és a userek még megvannak!

Így hát – még mielőtt teljesen megváltam volna a régi felállástól és AD-tól – készítettem egy-egy teljes dumpot az LDIFDE és a CSVDE eszközök segítségével.

```
C:\csvde -v -f c:\output.csv
```

Illetve,

```
C:\ldifde -v -f c:\output.txt
```

Az említett parancssori eszközökről röviden annyit, hogy az AD szöveges dumpjához, szerkesztéséhez, export és import operációkhoz használhatjuk őket. A régi villamosos viccel élve különbség kettőjük között az, hogy az LDIFDE „így hosszú” és a CSVDE meg „így széles”.

Az elsőben minden AD-objektum adata egymás után szerepel egy végtelenül hosszú listában egymás alatt. Mindenkinek minden adata és adatmezője szépen sorban. A CSVDE viszont jobban alkalmas emberi fogyasztásra mert ott az adatokat meghatározó mezők csak egyszer, egymás mellett szerepelnek az első vízszintes sorban. Így attól függően, hogy egy adott mező értelmezhető egy-egy objektumra: vagy tartalmaz adatot, vagy nem. Az információk objektumonként, egy-egy sorban helyezkednek el. Ez a forma egy Excel táblába beolvasva jól átlátható, kis ügyességgel szerkeszthető, kezelhető. Azt mindkét esz-

közről tudni kell, hogy ha az ember készít egy exportot, és megpróbálja akár azonnal, mindenféle változtatás nélkül visszatenni, nem fog sikerülni!

Az export ugyanis tartalmaz olyan mezőket is, amik nem szerepelhetnek az import során. Tehát ha mindent kiexportálunk, a szöveges szerkesztés elkerülhetetlen. *(A kézi szerkesztéssel a Q276382 számú MS KB cikk foglalkozik részletesen. Emellett az eszközök kapcsolóival meg lehet adni, hogy milyen objektumokat, melyik tárolóból, és melyik attribútumokkal exportálja ki. Ömleszteni egyszerűbb és gyorsabb, igaz, akkor felesleges adataink is lesznek.)*

Ritkán kell, de ha kell, rettentő hasznos, hogy ezek az eszközök a rendelkezésünkre állnak. Nekem például nem kellett kézzel újra létrehoznom a 48 usert meg a legalább ugyanennyi csoportot, meg egyéb – a userekre jellemző – beállításokat. Nem is említeztem úgysem a pontos nevekre, lévén, hogy nagyrészt külföldiekről van szó. De ha mindenki magyar nevű lett volna, akkor sem emlékeztem volna, hogy hány és milyen csoportot hoztam létre az elmúlt egy év során. Eldöntöttem tehát, hogy az exportok birtokában nekikezdek a teljes reinstallnak.

Így is tettem és az [**sbsrecovery**] címen elérhető SBS Server Recovery cikk szerint elkezdtem az ott ismertetett eljárást, bízva abban, hogy a mentés, ami a rendelkezésemre áll, tartalmaz minden olyan adatot, ami kell. A System State-et persze elfelejthettem ebben a speciális esetben. De miután vége lett ennek az egésznek, és ez a szerver visszakerült a rackszekrénybe, fogtam egy friss mentést és tesztképpen végigcsináltam lépésről-lépésre a dokumentumban ismertetett folyamatot egy másik gépen és a végén – szkeptikus hozzáállásom ellenére is – minden a legnagyobb rendben működött! SBS Restore igenis létezik! Csak jó mentés kell hozzá!

Visszatérve az eredeti szerverhez: tehát miután elkészült egy új – de nevében és jellemzőiben a régivel teljesen azonos – SBS 2000 install, felkerültek a frissítések és programok, elővettem a CSVDE eredményét, és egy kis szerkesztés után – aminek elkészültében nagy-nagy segítségemre volt főnököm tudása, türelme és kitartása – elvégeztem az importot, és lám, megjelentek az AD-ban az objektumok. Minden ott volt, ami kellett: Containerek, Userek, Csoportok.

Még egy nagy falat, az Exchange!

Nos, ebben az esetben, mivel az eredeti System State nem jött vissza, az Exchange szempontjából a pillanatnyi felállás teljesen új szerverként jelentkezett. Még akkor is, ha a neve, domainje ugyanaz volt, mint annakelőtte. Ez persze csak akkor gond, ha adatbáziszintű restore-t próbálunk. Azt ugyanis igencsak nehéz elfogadtatni az „új" szerverrel, hogy megszeresse egy „másik" szerver adatbázisát, logjait. Itt majdnem feladtuk, mert sehol nem találtunk olyan megoldást, ami hibás System State mellett lehetővé teszi az Exchange disasterrecovery telepítését.

Ekkor jött – sokadik álmatlan éjszaka után – a mentő ötlet: „Hiszen a Veritas Backup Exec-nek van Exchange Agent-je, azaz ő ért Exchange-ül." Nosza, szedjük elő azt, és mivel a userek, alias-ok, csoportok a neveket tekintve pontosan megegyeznek a régi állapottal *(a CSVDE import miatt)*, az individuális mailboxonkénti mentés szépen visszatehető mindenkinek. Ez ugyanis nem adatbáziszintű visszaállítás. Ő csak az adatokat, leveleket, és egyéb mailbox-beli dolgokat adja oda, amit az ESE szépen betesz a helyére az ő belső lelkivilága szerint. Még a beállításokat és a rule-okat is!

Persze azért ez sem volt ilyen egyszerű. A mailboxoknak ugyanis fizikailag is meg kell lenniük az adatbázisban, mielőtt így írhatnánk beléjük, illetve kell egy mentést készíteni az egészről, mielőtt használhatnánk az ilyen stílusú visszaállítást. Fizikai meglét alatt azt értem, amit: nem elég a szerveren megadni, hogy X.Y.-nak legyen levelesládája, mert az Exchange lusta, és addig nem csinálja meg, amíg erre ténylegesen nincs szükség.

A fizikai létrehozás egyszerű: be kell lépni klienssel. Ezután a levelesláda már célpontja lehet egy visszaállításnak, aminek végeztével azért adódott még egy probléma.

A domain régi tagjai – a felhasználói gépek – ennek az „új" domain-nak nem voltak tagjai. Így ki kellett venni kézzel minden egyes gépet, majd visszatenni, és az újonnan létrejött lokális user-profile-okba visszamásolni az adatokat a régiekből. My Documents, Desktop, Favorites, stb. De ezek után tényleg minden működött és egy félig használhatatlan mentés birtokában is kijelenthettük, hogy semmi nem veszett el azon az egy héten kívül, amíg minden állt ;-) !

> Ha van is mentésünk, vajon jó-e? Hihetjük azt is, hogy minden rendben. De egy próbát mindenképp megér, hogy biztosak is legyünk benne!

Minden régi adat megvan és minden használható!!! Ötven ember egyévnyi munkája, pénzügyi adatai, levelezése, dokumentumai. Szinte minden, amin eddig dolgoztak. Valóban nagy szó, hogy semmiről nem kellett lemondaniuk. Sőt, a régen tervezett víruskereső-motor frissítés helyett is mostmár a legújabbat tettem fel.

Időközben kiderült a probléma tényleges oka is. Valóban hardverhibáról volt szó, és természetesen a memóriák körül volt a gond*(!)*. Egy hibás modulban sorra módosultak az ott előforduló adatbitek, ezek ugyanígy kerültek kiírásra is a diszkekre. Szépen lassan – hibát hibára halmozva – összejött egy jó kis Systemcrash! De mint láthatjuk, semmi sem menthetetlen, csak kis kitartás és elszántság kell hozzá. Végső konklúzióként megemlíteném a teljes mentés fontosságát és a Veritas Backup Exec software hasznos kiegészítését, az Exchange Agentet. Enélkül ugyanis nem lehetséges a mailboxok egyenkénti mentése és visszaállítása. A beépített NT-Backup sem tud ilyesmit.

Van mégvalami, ami sokat segít egy-egy ilyen helyzetben, ez pedig a kezdők szerencséje. Ugyanis élesben még egyikünk sem használta a CSVDE tool-t, így nem is kötötte a kezünket semmiféle előítélet, tapasztalat. Abban bíztunk csak, hogy az adott szituációban már semmit sem veszíthetünk. Rengeteg utánaolvasás és persze a kényszerhelyzet szülte feszült légkör eredményeként a jövőben biztos kézzel nyúlok majd ezekhez a ritkán használt eszközökhöz. Jótanácsként javaslom, ha lehetőségünk van rá, tegyünk el egy-egy mentést az utókornak legalább háromhavonta, mert ha nekem ilyenem lett volna, ez a cikk nem lett volna ilyen hosszú…

Füzesi Szabolcs
fuzesisz@osi.hu
MCSA

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

Gyógyszeresdoboz IV.

Windows 2000 Server Resource Kit

A Network Management Tools II., azaz a sorozat előző részében taglalt, felhasználókkal, csoportokkal és megosztásokkal kapcsolatos eszközök átvilágítása után, most tekintsünk bele egy másik világba.

A TCP/IP udvartartásába tartozó szolgáltatások *(DNS, DHCP, WINS)* mindennaposan használt technológiák, amelyek alapos, mélyreható ismerete nélkül elképzelhetetlen egy-egy rendszer tervezése és üzemeltetése. A kiszolgáló- és kliensalkalmazások egyaránt erőteljesen támaszkodnak ezekre a megoldásokra, így emiatt egy hálózat stabilitása, megfelelő hatásfokú működése is múlhat rajtuk *(különösen igaz ez egy Windows 200x tartományban, ahol mint tudjuk nincs is élet pl. DNS nélkül)*. Ezen ismeretek hiányában szembesülhetünk még egy gonddal: elakad*(hat)* a törvényszerűen legrosszabb pillanatban megtörténő problémák gyors felismerése illetve megoldása. Persze, van egy Murphy törvény, vagy inkább egy törvényi folyomány, amely szerint „A problémamegoldás titka az, hogy meg kell találnunk azt az embert, aki megoldja a problémainkat", de én inkább szeretem azt hinni, hogy a tudás és a megfelelő segédeszközök fontosabbak. Az utóbbiakról – miután túlestünk az érdemtelenül hosszú bevezetésen – lesz most szó.

Dhcploc.exe

Szinte minden MCP tesztben visszaköszön az a Windows 2000 DHCP szerverekkel kapcsolatos kérdés, hogy ha minden beállítás korrekt és mégsem működik a szerver, mi a hiba. A hiba az újdonság, mégpedig az, hogy a DHCP szervert a Windows 2000 Server-től kezdve hitelesíttetni kell a címtárral. Amennyiben az Enterprise Administrator csoport tagjaként telepítettük, a DHCP MMC-ben ezt egy egyszerű kattintással a szerver helyi menüjében megtehetjük *(Authorize)*. Kis idő elteltével nagyszerű vizuális élményben is részesülhetünk, hiszen az eddig piros színben tündöklő kis nyíl zöldre vált a szerver ikonján, ha rendben lement a folyamat. De miért van erre szükség? A kalóz *(rogue)* DHCP szerverek kiszűrése miatt, amelyek jelentősen bomlaszthatják jól működő hálózatunk szerkezetét, abban az esetben ha valamelyik „kollégánknak" direkt avagy véletlenül sikerült egy saját, mondjuk nem túl korrekt beállításokkal rendelkező DHCP szervert üzembehelyezni és aztán magához csalogatni a klienseket. Ilyen illegális DHCP szerverek felfedezésére használható *(NT4 Serveren is, ahol nincs hitelesítetés)* a DHCP Server Locator Utility azaz Dhcploc.exe.

Az eszköz csak a saját alhálózatán tud DHCP üzeneteket elfogni és működéséből adódóan nem futatthatjuk magán a legális DHCP szerveren sem **[Q186462]**. Ha megfelelően paraméterezve elindítjuk, szépen elkezdi a hálózat lehallgatását, figyeli a speciális DHCP csomagokat és ki is írja ezeket automatikusan. Futás közben három billentyűvel adhatunk ki parancsokat *(ezt egyébként sehol nem jelzi)* „d" mint Discovery *(azaz ajánlatkérés, ha másra nem, erre majd csak felfigyelnek a DHCP szerverek)*, „q" mint Quit és „h" mint Help. Abban az esetben, ha a program kalóz DHCP szervert talál, sípol és egy figyelmeztetést is küld. A következő példa alapján kideríthetünk egy s mást a szintaxisról is:

```
dhcploc -p -a:"alert.txt" -i:300 10.0.0.204
↳ legalis.txt
```

Látható pl. hogy a „**-p**" és a parancs végén a „**legalis.txt**" használata az általunk megadott, korrekt DHCP szerver*(ek)*et már eleve kiszűri a vizsgálatból. Az „**-a**" kapcsoló után lévő szövegállományban azt a címzettet kell megadni, ahová a figyelmeztetés megy, az „**-i**" az automatikus vizsgálat intervalluma másodpercben, az IP cím pedig az alkalmazást futtató gép címe. A kimenet részlete lehet például valami hasonló:

```
17:35:58 (IP) 10.0.0.210 OFFER (S)10.0.0.226 ***
17:35:58 (IP) 10.0.0.79 ACK (S)10.0.0.205
17:36:13 (IP) 10.0.0.79 ACK (S) 10.0.0.205
17:36:25 (IP) 10.0.0.141 OFFER (S) 10.0.0.205
17:37:10 (IP) 10.0.0.72 OFFER (S)10.0.0.226 ***
```

Az oszlopokban az időpont, a kliens IP címe illetve az ajánlat/jóváhagyás típusa és az ajánló szerver IP címe látszik. A három csillaggal jelzett szerverekkel van a baj, valószínűleg megértek a likvidálásra!

Dhcpobjs.exe

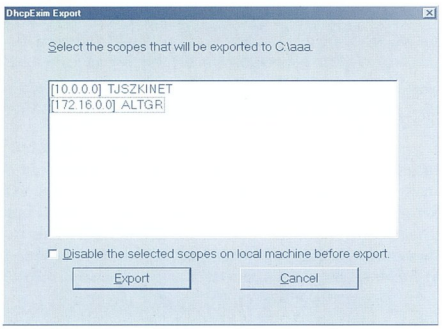
A már többször emlegetett Windows 2000 Server Support Toolsban létezik egy dhcpadm.exe, amellyel parancssorból lehet irányítani a DHCP szerver általános és alapvető funkcióit. De nem mindent. A „minden", ami kiegészíti a dhcpadm.exe lehetőségeit, az a Resource Kitben szereplő DHCP Objects csomag. Ennek feltelepítése után lehetővé válik alkalmazásokból illetve VBScript és JavaScript szkriptekből a DHCP szerver teljeskörű távoli üzemeltetése és felügyelete. Ez a komponens alapból nem települ fel a Resource Kittel, külön kell kicsomagolni az .exe-ből, majd a szükséges dll állományt is nekünk kell manuálisan regisztrálni a regsrv32.exe-vel. A csomagban érkezik még egy Dhcpobjs.chm nevű HTML formátumú súgó, amely nagyon részletes és tele van példákkal.

Dhcpexim.exe

Ha már a vizsgakérdéseknél tartunk, bizonyára sokan emlékeznek arra is, hogy szintén sűrűn sor kerül a DHCP adatbázissal kapcsolatos mentési, mozgatási műveletek firtatására, ezért célszerű ismerni a manuális módszert. Ennek lépései dióhéjban: a regisztrációs adatbázis idevágó kulcsának exportja, majd a System.mdb állomány átnevezgetése és végül registry import abból a célból, hogy egy másik *(vagy akár ugyanazon)* gépen a sikeres visszatöltés után változatlan állapotot tudjunk prezentálni.

Ez a módszer jól jön abban az ismert szituációban is, ha szervert váltunk és előtte már kismillió tételt bejegyeztünk *(illetve automatikusan bejegyzésre került)* ebbe az adatbázisba. A teljes művelet **[Q130642]** kissé macerás dolog, de muszáj, mert a DHCP szerverben nincs mentési lehetőség és az NTBackupnak sincs ilyen opciója.

De hogy mégse legyen annyira muszáj, ebben az esetben is segít rajtunk a Resource Kit, pontosabban a „nagy" Resource Kit szűkített, letölthető változata, amelyben szerepel egy erre a célra tökéletesen használható és ezért igen szimpatikus eszköz a DHCP Database Export Import Tool, amely innen **[dhcpexim]** beszerezhető.



Választhatunk a hatókörökből az export során

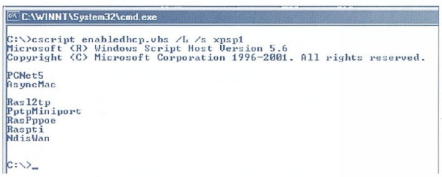
Az eszközt nagyon egyszerűen csatasorba tudjuk állítani, gyakorlatilag egy varázsló végigvezet az export ill. az import lépésein. A telepítés után a Resource Kit mappában találjuk meg *(akkor is, ha nincs meg a teljes Resource Kitünk)*. A „varázslás" során kiválaszthatjuk a menteni kívánt hatókört, és természetesen nemcsak a bejegyzett címeket, hanem az egyéb beállításokat *(DNS szerver(ek), WINS szerver(ek), átjárók, stb. címei)* és a rezervációkat is elmenti. Vegyük figyelembe, hogy a varázsló leállítja a DHCP szolgáltatást, amíg az export/import megy!

Enabledhcp.vbs

Újabb okosság: ennek az apró kis szkriptnek a segítségével parancssorból, távolból lehet reszetelni egy-egy gép manuális TCP/IP beállításait és aztán a DHCP szerverre „kötni". Kényelmes dolog a szerverről futtatva átállítani így a kliensgépek tucatjait, de akár a laptopos ügyfelek számára is készíthetünk egy parancsikont, amire a felhasználó csak rábök és máris bent van a hálóban, látja az összes - a DHCP szerverben beállított - fontos hálózati szolgáltatást nyújtó kiszolgálót. A szintaxis két része oszlik, ám mindkettőnél a cscript.exe-vel kell futtatni a szkriptet *(az xpsp1 a távoli gép neve)*:

```
cscript EnableDhcp.vbs /L /S xpsp1
```

Az első példában az **/L** kapcsolóval tudjuk lekérdezni a távoli gép *(összes)* hálózati kártyáján futó szolgáltatásainak nevét. A **/S** kapcsoló mindig a kliens gép nevét feltételezi, azaz annak a „Server" szervizét. Az így kiadott parancsra egy hasonló listát kaphatunk, mint az alábbi képen.



A kliens hálózati kártyáján futó szolgáltatások

Ezek közül ki kell választanunk azt, amelyikkel a helyi hálózathoz csatlakozunk *(jelen esetben a PCNet5 > vmware)*. Ezt az **/A** kapcsoló után kell megadnunk. Az **/U** és az **/W** kapcsoló magáért beszél. Az eredmény pedig lementhetjük az **/O** kapcsoló után megadott állományba.

```
EnableDhcp.vbs /A PCNet5 /S xpsp1 /U Administrator
/W password /O c:\enabledhcp_log.txt
```

Winscl.exe

A kérdésre, hogy kell-e nekünk a WINS *(Windows Internet Name Service)* – azaz DNS helyettese illetve a Windows 9x/NT operáció rendszerek illetve az ezekre a platformokra írt alkalmazások esetén az elsőszámú névfeloldást segítő szolgáltatás egy TCP/IP-vel működő helyi hálózatban – nem könnyű válaszolni. Ha teljesen Windows 2000 Server kompatibilis klienseink vannak, és egyáltalán nincs szükségünk a problémás NetBIOS névfeloldásra, a redundancia miatt még mindig jó lehet a DNS helyett. De ha vannak még pl. Win9x/NT klienseink is *(és még mindig rengeteg helyen így van)*, nem sok választási lehetőségünk van, szinte muszáj használni.

Ha pedig kell a WINS, akkor szükség lehet a parancssori WINS Administration Toolra is, amellyel ellenőrizni tudjuk a WINS forgalmat, valamint megvizsgálhatjuk a szépen hízó adatbázist is. Az eszköz interaktív üzemmódban működik mint pl. az nslookup, de mégsem teljesen úgy: az elindítás után mindig látható a teljes parancskészlet illetve ezek 2-4 betűs rövidítése. Képes rábírni a WINS szervert olyan műveletek elvégzésére mint pl. a replikáció, a takarítás *(scavening)*, rekordok regisztrálása, törlése és lekérdezése az adatbázisból, a mentés és/vagy visszaállítás vagy éppen az inkonzisztencia vizsgálat.

A WINS és a DNS témakörével kapcsolatos további Resource Kit segédeszközök tallózása következő számban folytatódik.

Gál Tamás
MCSE 2000
gtamas@tjszki.hu

Megállás nélkül

99,999%-os rendelkezésreállás Windows operációs rendszerrel?

A cél az „ötkilences" rendelkezésreállás. Évente 5 perc leállás. Ilyenkor aztán beindul az agyalás: Hogyan? Farkasokkal táncolunk, clustert építünk és fürtözünk? Ide ez már nem elég. Ez már egy más kategória, itt a fürt hamar elvérzik. Ide Stratus kell!

Mi a baj a fürtökkel?

Amíg szőlőről van szó, semmi. Finom bor készül belőle. Az informatikában azonban más a helyzet. Mi a fürt?
Összekapcsolt gépek halmaza, melyen erőforrásokból álló csoportokat definiálunk. Az erőforrások függőségi viszonyban állhatnak egymással. Ha bármiféle probléma van, egy erőforrás elhasal, majd a kapcsolatok függvényében a többi is, ez kiváltja az adott csoport vándorlását. A futtatott alkalmazás **átköltözik** egy másik gazdagépre. A beállított szabályoknak megfelelően persze vissza is tud sétálni. Egyszerű. Valóban?
Itt szeretnék eloszlatni egy félreértést: fürtöt nem lehet venni. Fürtöt csak **építeni** lehet! *(Aki rakott már össze fürtöt, az tudja, hogy miről beszélek.)* Elő kell kapni a fürttéglákat *(2-3-4 szervergép, megfelelő eszközökkel)* és a fürtmaltert *(2-3-4 szerver licensz + 2-3-4 alkalmazás licensz, tipikusan az „űrhajó" változat)*, és lehet kezdeni az építkezést. Egy kritikus feladat kiszolgálására hivatott fürt építése gondos *(és időigényes)* tervezést, tesztelést igényel, és az implementáció is meglehetősen összetett feladat. A kialakított fürt üzemeltetéséről nem is beszélve. Mi szükséges ehhez? Rengeteg erőforrás *(szakember, idő, pénz)*. Az első nagy koppanás – szerencsés esetben a felmérés, tervezés során - akkor következik be, **ha a kritikus alkalmazás nem fürtözhető** *(nem cluster-aware)*. Ilyenkor jöhet az erőlködés, hogy „de mégis". Nem javaslom. A valódi probléma azonban magában az elképzelésben van. Mi a fürt célja? A **helyrehozás** *(recovering)*! Na itt bukik meg az „ötkilences"... Figyeljünk csak: A **hiba bekövetkezése után**, a fürt gondoskodik a kritikus szolgáltatás *(alkalmazás)* **helyreállításáról**. A helyreállítás időt igényel, és a kritikus szolgáltatás ez alatt szünetel.

Az ötlet

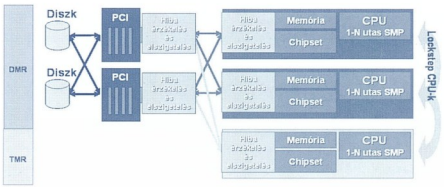
Az ötlet egyszerű. Készítsünk olyat, ami „nem romlik el". Itt is igaz – az orvosok által gyakran ismételgetett - bölcsesség: a „megelőzés sokkal hatékonyabb, mint az utólagos kezelés". Meg is érkeztünk a Stratus filozófiájához: **a leállást elkerülni kell, és nem minimalizálni!**

Stratus

Az elképzelés működik. A Stratus ftServer hibatűrő szerverek 99.999% rendelkezésre állást biztosítanak a Microsoft Windows 2000 alapú rendszerek számára. Mit jelent ez a gyakorlatban? Egy ftServer két nagyságrenddel jobb rendelkezésre állást nyújt, mint egy WINTEL alapú kiszolgáló fürt! Szépen hangzik, de hogyan lehetséges?
Nem is olyan régen Bill Gates meghirdette a Trustworthy Computingot. Felismerte, hogy egy rendszert nem lehet **utólag** biztonságossá tenni. Valódi eredményt csak úgy lehet elérni, ha a biztonság a legelső gondolattól kezdve szerves része a fejlesztésnek. Secure by design.
Hasonló a helyzet a megbízhatósággal és rendelkezésreállással is. Amikor hagyományos gépekből építünk fürtöket, az egyébként „megbízhatatlan" gépekből próbálunk megbízható rendszert építeni. Több-kevesebb sikerrel.
Ha valóban megbízható rendszert szeretnénk építeni, az alapoknál kell kezdeni, és már a tervezés első lépésétől kezdve a megbízhatóságnak kell lennie a fő célnak. Ezt a módszert követte a Stratus is. Reliability by desing.
Az eredmény egy egészen speciális hardverarchitektúra lett.

Hibatűrő hardver

Egy Stratus belül alig hasonlít a hagyományos szerverekre. Minden komponens duplán, vagy triplán található meg benne. Dupla, tripla CPU, memória, merevlemezek, PCI kártyák, ventillátorok, tápegységek speciális hibaérzékelő és elszigetelő rendszerrel ellátva. Nincs „single point of failure".



Hibatűrő architektúra

Speciális HAL

A komponensek többszörözése önmagában még nem lenne elegendő a sikerhez, az operációs rendszert is fel kell készíteni a hibatűrő architektúra kezelésére. Ezért a Stratus - a Microsofttal együttműködve – speciális HAL-t *(Hardware Abstraction Layer)* készített szervereihez. Így a szerveren futó operációs rendszer, és főleg a futtatott kritikus alkalmazás nem érzékeli az egyes komponensek meghibásodását. Mit jelent ez? Nincs leállás. Gyakorlatilag alkatrészenként a komplett gépet ki lehet cserélni menet közben, miközben az alkalmazás folyamatosan működik. A fürtökre jellemző átállási időt is el lehet felejteni, mert nincs.

Driverhibák nélkül

A hibás driverek előkelő helyen állnak minden üzemeltető feketelistáján. Számos probléma és összeomlás valódi oka egy hibás eszközmeghajtó. A becslések szerint az NT-nél az új-

raindulások, és rendszerhibák több mint 30%-át hibás eszközmeghajtó okozta. A Windows 2000 esetében a Microsoft sokat szigorított az eszközmeghajtók tesztelési és minősítési eljárásain, de a hibás eszközmeghajtók ennek ellenére még mindig nagyon sok problémát okoznak.

A Stratus ennek ismeretében a legnagyobb körültekintéssel fejleszti ki és teszteli a szervereiben használt eszközök meghajtó programjait, így minimalizálva ezt a jelentős kockázati tényezőt.

Beépített szerviztechnológia

A beépített szerviztechnológia a folyamatos működés egyik kulcsa. Ezek a szoftver- és hardverelemek folyamatosan ellenőrzik a kiszolgáló működését. Ha egy komponens meghibásodik, azt azonnal elszigetelik, és rendszer automatikusan bejelenti a hibát a szervizközpontba. Amennyiben a hiba megoldható a kiszolgáló management kártyáján keresztül, a szervizközpont on-line beavatkozik, amennyiben nem, a helyszínen kicserélik a hibás komponenst. Természetesen ez a felhasználók számára teljesen láthatatlan, mert a szolgáltatások ez idő alatt is folyamatosan működnek. Miért fontos ez a szerviztechnológia? Mondok egy példát: gondoljuk csak a mindenki által ismert RAID technológiára. Ennek a – manapság szerencsére már mindenki számára elérhető – technológiának az a lényege, hogy a diszk alrendszer valahány *(tipikusan 1)* diszk meghibásodását adatvesztés nélkül elviseli. Ezt a rendszer valóban el is viseli. Meghibásodik egy diszk, a rendszer ugyan lassul, de az üzemeltetés esetleg nem vesz észre semmit, hiszen minden működik. Akkor lesz igazán baj, ha a következő diszk is felmondja a szolgálatot. Két lemez meghibásodását *(konfigurációtól függően)* a rendszer nem képes elviselni, itt a **hiba**, a **leállás**. Ha az első hibás lemezt azonnal kicserélték volna, a rendszer visszanyerte volna hibatűrő képességét és működhetne tovább. Tehát az egyébként kifogástalanul működő technológia is hatástalan megfelelő jelzés, üzemeltetés és felügyelet nélkül.

Üzemeltetés

Szép feladat rendszereket építeni, de az igazán kemény feladat a rendszermérnökök által összerakott és dokumentált rendszer életben tartása. A Stratus esetében ez meglehetősen egyszerű, hiszen – a fürtökkel ellentétben – egy operációs rendszer fut rajta, és az alkalmazás is csak egy példányban működik. Nem kell fail-over scripteket írni, tesztelni, és aggódni a fürt aktuális gyermekbetegségei miatt. A kifinomult hibaérzékelő és izoláló rendszer gyakran a tényleges meghibásodás előtt képes a mért paraméterek segítségével előre jelezni a problémát, értesíteni az üzemeltetést és a szervizt.

Uptime meter

Szép gondolatok, de ismét felmerül a kérdés: mit jelent ez a gyakorlatban? Az eredményt bárki megtekintheti, a Stratus weboldalán *(www.stratus.com)* lévő uptime meter a jelenleg menedzselt ftServer-ek rendelkezésreállását jelzi *(a telepített szerverek 80%-a)*. A cikk írásának időpontjában a számláló a következő értéket mutatta:

Hová?

Szükséges ekkora rendelkezésre-állás? Ez nem technológiai döntés. A választ a védeni kívánt alkalmazás/ok ismeretében az üzleti döntéshozóknak kell meghoznia. Napjainkban az üzleti szempontból kritikus alkalmazások elérhetőségével kapcsolatos elvárások egyre magasabbak. Egyre kevesebb szervezetnél fogadható el az ügyviteli rendszer, az elektronikus levelezés, vagy az adatbáziskezelő 8 órányi leállása.

A múltban ötkilences rendelkezésreálláshoz sok millió dolláros űrhajók beszerzésére volt szükség, többnyire teljesen speciális operációs rendszerrel és alkalmazásokkal.

A jó hír a következő: ez a megbízhatósági szint elérhető szabványos Windows platform használata mellett, a megszokott WINTEL árszinten is.

Détári István
istvan.detari@getronics.hu
A szerző a Getronics *(Magyarország)* Kft. tanácsadója
MCSE, MCSA, MCDBA, CCNP

A DHCP rejtett szépségei – IV.

Egy végső lendülettel megbirkózunk a DHCP utolsó nagy fogalomkörével, vagyis az osztályokkal és opciókkal. A cikksorozat második részében már megemlítettük a legfontosabbakat, de túlságosan nem merültünk el a részletekben. Most viszont épp ezt fogjuk tenni, hogy megértsük, milyen eszközeink vannak az ügyfelek finomhangolására, s egyáltalán: mi mindenre képes a DHCP...

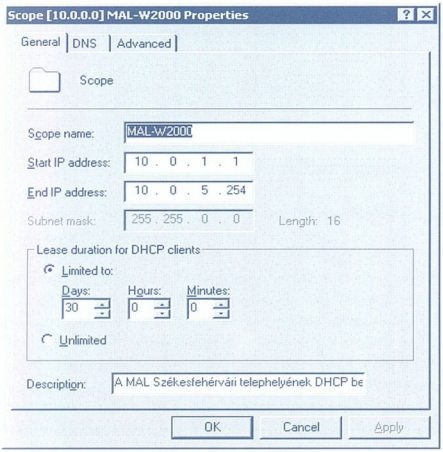
Az opciókról általában

Az opciók lényegüket tekintve paraméterek, amelyeket az ügyfél megkap, feldolgoz vagy éppenséggel elvet. A szabványt úgy alkották meg, hogy a DHCP-csomagokban opciók tömkelege, egészen pontosan legfeljebb 312 byte *(oktet)* vándorolhat a kiszolgálóról a klienshez. Csoportosításukkal világos képet alkothatunk, hogy mikor kell beállítani egyet-egyet, s mikor nem.

Többféle rendezés is elképzelhető. Technikai értelemben léteznek információopciók *(information options)* és protokollopciók *(protocol options)*. Az információopciókat explicit módon be kell állítani, hogy érvényre jussanak. Az alábbiak mind ennek a csoportnak tagjai:

3	Router
6	DNS kiszolgáló
15	DNS domain név
44	WINS kiszolgálók
46	WINS/NetBIOS node típus
47	NetBIOS Scope ID stb.

A protokollopciókat ezzel szemben csak implicit módon lehet megadni a bérlettartomány létrehozásával és konfigurálásával, vagyis egy scope létrehozása részben DHCP-opciók megadása, még ha ezt el is rejti a felhasználói felület.



A bérletidő megadásával az 51-es, 58-as és 59-es opciót is beállítjuk. A subnet mask opciószáma 1.

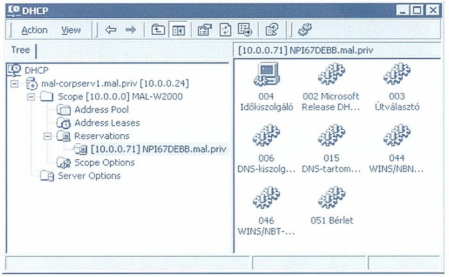
Az alább felsoroltak mind a protokollopciók közé tartoznak:

51	Bérletidő
53	DHCP-üzenettípus
55	Igényelt paraméterlista
58	Megújítási idő *(Renewal time)*
59	Címkérési idő *(Rebind time)*

Létezik másfajta csoportosítás is, ekkor az alapvető jellemző az a hatókör, amelyben a paraméter felhasználható. Ez alapján megkülönböztethetünk:

1. Szerver avagy default global opciókat
2. Scope opciókat
3. Class opciókat, amelyek lehetnek User class és Vendor class opciók
4. Kliens opciókat, amiket lefoglalt *(reserved client)* opcióknak is hívnak.

Mindez persze nem jelent pontos egymásba ágyazódást. A szerver-scope-client opciók még jól követhetők a konzolon is, ám a user class és a vendor class opciók a szép egymásba ágyazódást keresztülszelik, mert elvileg bárhol előfordulhatnak.



A szerver, a scope és a client opciók egymásba ágyazódnak

Az erősorrendet ezzel együtt nem nehéz felállítani. A leggyengébb opció-csoport a szerverhez kötődik. Ezek minden bérlettartományban érvényre jutnak, hacsak a scope-ban definiáltak felül nem írják őket.

A bérlettartományhoz rendelt opciók a leggyakoribbak. Minden ügyfél megkapja őket, és amennyiben érti, fel is dolgozza azokat. Mit is jelent ez pontosan? Nos, előfordul, hogy egy ügyfél olyan paramétert is kap, amely őt nem érdekli. Például egy Windows for Workgroups 3.11-es ügyfelet egyáltalán nem foglalkoztatja a hálózatban fellelhető NIS szerverek IP-címe

(Option 41), ugyanakkor egy Linux rendszer számára meghatározó információról lehet szó. A WfW 3.11.-be egyszerűen nem programozták bele valamennyi opció feldolgozását, így az csak a számára relevánsakat fogja megérteni. S melyek ezek? Ezt már szintén érintettük korábban, de nem árt ismételni. A Microsoft ügyfelek által használt paraméterek a következők:

01	Subnet mask
03	Router
06	DNS Servers
15	Domain Name
44	WINS/NBNS Servers
46	WINS/NBT Node Type
47	NetBIOS Scope ID
51	Lease Time
58	Renewal Time *(T1)*
59	Rebinding Time *(T2)*

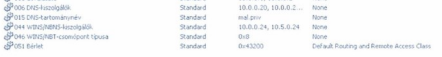
Inverz szedéssel jelöltem a protokollopciókat – ezeket a rendszer mindenképp kiadja, ahogy azt már kicsit korábban láttuk. Visszatérve az erősorrendhez: a szerveropciókat felülírhatják a scope-opciók, azokat pedig a kliensopciók. Ezt a sort színesítik némiképp az opcióosztályok.

Opcióosztályok *(option classes)*

Az osztályokat azért hozták létre, hogy további finomításokat végezhessünk a címkiosztási rendszerünkön. Nem kötelezően, de általában a scope-on belül definiáljuk azokat a beállításokat, amelyeket egy meghatározott osztályhoz szeretnénk rendelni.

Aki figyelmesen megnézi a bérlettartományhoz tartozó paraméterek részletes listáját, láthatja, hogy az oszlopok között szerepel egy „vendor" *(szállító)* és egy „class" *(osztály)* elnevezésű is. Egy hétköznapi opció, például a 15-ös szállítója „standard", osztálya pedig „none".

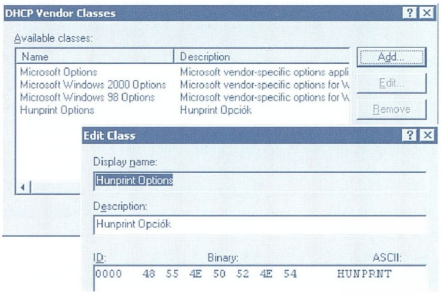
Ebből az következik, hogy egy opcióhoz rendelhetünk szállítót és osztályt is, sőt akár mindkettőt egyszerre. Az alábbi ábrán látható, hogy a 002-es a „Microsoft Options" kategóriába tartozik, a 051-es pedig a „Default Routing and Remote Access" osztályba.



Szállítók és osztályok alkalmazás az opciók között

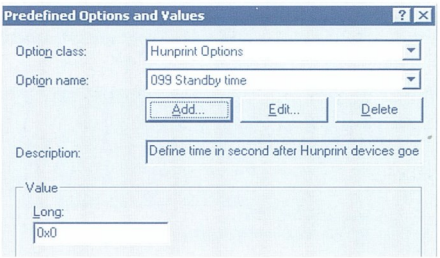
Mi is a fenti konfiguráció értelme? A 002-es opció érvényre jut minden „Microsoft Options" szállítói kategóriába eső ügyfélnél. Melyek ezek? A Windows 98 és annál frissebb, valamint a Windows 2000 és annál frissebb MS rendszerek. Az NT4 miért nem? Mert a szabvány, amely az osztályokat és a szállítói opciókat definiálta 1997-es keltezésű, tehát az NT4 megjelenése után alkották. És honnan tudja egy Windows 2000-es ügyfél, hogy ő ebbe a szállítói kategóriába tartozik? Ezt egyszerűen tudja, mert a szállítója beleprogramozta. Bármely szállító *(Pl. Xerox, Cisco, Motorola, Ericsson stb., stb.)* definiálhat saját opciókat és saját szállítói osztályokat, amelyeket azután a DHCP szolgáltatás ki tud osztani a megfelelő szállítóazonosítóval rendelkező ügyfelek számára.

Nézzük meg mindezt egy fiktív – ám mégis lehetséges példán. Tegyük fel, hogy hálózati nyomtatók gyártására szántuk el magukat, a cég márkanevéül pedig a nem túl fantáziadús, ám jól csengő HunPrint nevet adtuk. A nyomtatónkat felkészítettük arra, hogy egy DHCP kiszolgáló segítségével távolról konfigurálni lehessen azt az időt, ami után, ha használaton kívül van a gépünk, automatikusan energiatakarékos üzemmódba *(standby)* vált. A nyomtató operációs rendszerébe beágyaztuk a megfelelő kódot, és tudattuk vele, hogy a HUNPRNT szállítói osztály tagja. Ezek után ezt az osztály létre kell hoznunk a DHCP kiszolgálón is. A konzolon kattintsunk a jobb egérgombbal a szerveren, majd válasszuk a „Define Vendor classes..." pontot. A három, előre definiált osztályt láthatjuk, ezt a listát kell kiegészíteni a sajátunkkal.



Szállítóosztály készítése

Ezután a fenti context-menüt újra előcsalogatva definiálhatunk egy új beállítást a „Set Predefinied Options..." funkció segítségével, sőt, még alapértelmezett értéket is adhatunk neki.



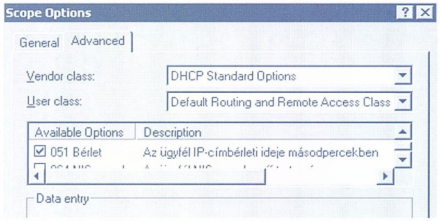
Definiáltunk egy saját opciót a nyomtatóinkhoz

Ha a fenti opciót kiajánljuk egy bérlettartományban, minden eszköz, amely a szállítói osztálynak tagja *(vagyis a nyomtatóink)*, gond nélkül alkalmazni fogja.

Persze mi nem fogunk nyomtatógyártásba kezdeni, de lehetséges, hogy a fenti neves gyártók egyike-másika élni fog a lehetőséggel, hogy precízebb beállításokkal lássa el a gyártmányait a DHCP szolgáltatás segítségével.

A fenti példával analóg módon lehetséges user class és user class option létrehozása is, ám ekkor alkalmazhatunk egy további trükköt is.

A beállításánál az „Advanced" fülre kattintva kiválaszthatjuk, hogy melyik szállítói osztályból szeretnénk opciót beállítani.



Előre beállított User class opció alkalmazása

Az opciók többsége a „DHCP standard Options" szállítói osztályba tartozik. Az azonban, hogy mely user classba soroljuk, rajtunk múlik. Így lehetséges, hogy egy opciót másik user classba sorolunk, mint az alapértelmezett. Melyikbe sorolhatjuk? A Windows 2000 DHCP implementációja 3 előre definiált user classt ismer:

- Default user class
- Default remote access class
- Default BOOTP class

Az elsőbe tartozik minden ügyfél, aki nem definiált magának user classt. A másodikba tartoznak a Microsoft ügyfelek, ha betárcsáznak, a harmadikba pedig a BOOTP ügyfelek.
Ezek kívül természetesen magunk is létrehozhatunk user classt. Ha például szeretnénk, hogy minden gép, amely a humánpolitikai osztályon működik, egy alternatív alapértelmezett átjárót használjon, akkor létre kell hoznunk egy HUMAN user classt, be kell állítanunk egy 03-as opciót hozzá, végül konfigurálnunk kell valamennyi gépet a humánpolitikai osztályon. Ezt a régi ismerős ipconfig paranccsal végezhetjük el

```
P:\>ipconfig /setclassid "Local Area Connection"
human

Windows IP konfiguráció

Az adapter (Local Area Connection)
osztályazonosítónak beállítása sikeresen
megtörtént.

P:\>
```

Sajnos jelenleg egy adott host csak egyetlen user classnak lehet tagja, mivel a szabvány csak egyetlen ASCII stringet használ az ügyfelek azonosítására. Ezért ha van egy „human" osztályunk és egy „mobil" osztályunk, a mindkét osztályba tartozó gépekhez egy külön osztályt kell definiálnunk, például „human-mobil" néven.

A szállítói és a user class közötti különbséget jól szemlélteti a fenti két példa. Bár mindkét módszer az ügyfelek szelektív konfigurálására szolgál, az egyik a gyártóspecifikus konfigurációs paraméterek továbbítására tervezték, míg a másik a rendszergazdák által valamilyen kritérium szerint felállított ad-hoc csoportok elkülönített paraméterezését segíti.
Az ábrákon egyébként nem véletlenül szerepelt többször is a Default Routing and Remote Access Class. Ennek az osztálynak a segítségével elérhetjük, hogy az RRAS ügyfelek csak rövidebb ideig kapjanak DHCP címeket úgy, hogy a 051-es opció segítségével lerövidítjük a bérletidőt.

Miután áttekintettük mindazt, amit az opciókról tudni érdemes, egy elméleti kérdést meg kell válaszolnunk: Vajon miért nem alkalmaz szélesebb körben opciókat a Microsoft az operációs rendszereiben? A user class azonosítókkal például LPR nyomtatókat lehetne megadni egy-egy osztálynak, vagy be lehetne állítani egy idő-kiszolgálót stb. stb. Ehelyett a világelső szoftvercég ismét letér az Internetes járt útról és csoportházirendeket fabrikál *(ahol például éppúgy meg lehet már adni az alapértelmezett átjárót, a DNS szervereket, a DNS prefixet és lehetne még sorolni).*
A válasz kézenfekvő: a DHCP szabvány 312 bájtnyi adatot *(paramétert)* továbbíthat. Ez bizony egy tisztességes házirendhez kevés. Ráadásul bonyolult adatstruktúrák sem közvetíthetők a DHCP segítségével. A testreszabhatósága, az ügyfelek megkülönböztetése sem túlságosan kifinomult *(pl. az Active Directoryhoz képest.)* Summa summára: a csoportházirendnek van helye a nap alatt.
És akkor hosszútávon DHCP-re nem lesz szükség? Hát, hosszú távon ott van ugye az IPv6, ami sokkal inkább képes konfigurálni magát, mint az IPv4, tehát a DHCP súlya valamennyire csökken majd. Az IPv4 világban azonban várhatóan mindig is megmarad, mert az eredeti funkciójára, vagyis az IP címek kiosztására továbbra is ez a legalkalmasabb módszer. Ami pedig a Microsoft ügyfélrendszereket illeti, egyre több DHCP-opciót fogadnak el, s mire beköszönt az IPv6-korszak, talán mindet ismerni fogják – de addigra kihal a DHCP.
Szerveroldalon ugyanakkor várható, hogy az újabb verziók követni fogják a DHCP szabványok változását, hiszen nemcsak Microsoft ügyfeleket kell kiszolgálniuk, s más rendszerek esetén – ha nincs házirend – marad a DHCP.

Lepenye Tamás, MCSE 2000
lepenyet@mal.hu

Felhasznált és ajánlott irodalom

Q240247 - How to Create a New DHCP User or Vendor Class
Q298844 - DHCP Server Handles the Global User Class Lease Option Incorrectly
Q266675 - Microsoft DHCP Vendor and User Classes
Windows 2000 Resource Kit – TCP/IP Core Networking Guide

Internet Information Services 6.0, IV. rész

Levelezőszolgáltatások a Windows Server 2003-ban: a POP3

Régóta vártunk arra, hogy a levelek fogadásáért és kiküldéséért felelős SMTP-szolgáltatás mellett a felhasználók postafiókját és az ahhoz való hozzáférést biztosító POP3 is megjelenjen a palettán. A Windows Server 2003-ban végre itt van – kicsit butuska, kicsit egyszerű, de a mienk :-)

A POP3 protokoll

A POP3 protokoll segítségével a felhasználók letölthetik a levelező-kiszolgálóra, részükre beérkezett, és ott – saját postafiókjukban - tárolt leveleket. A postafiókok karbantartása a POP3-kiszolgáló feladata. A felhasználó a levelek letöltése után törölheti, vagy akár a kiszolgálón meg is tarthatja a postafiókban tárolt leveleket. A POP3 protokollt az **[RFC1939]** definiálja, magyar leírását az Olvasó megtalálhatja a tech.net magazin 2000. decemberi számában, illetve a **[tudpop3]** címen.

A POP3 szolgáltatás

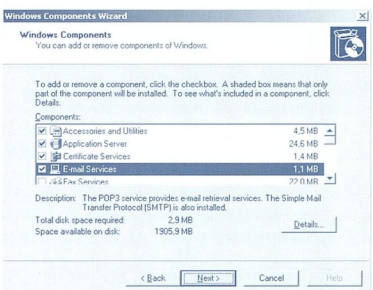
A POP3-kiszolgáló tehát lehetővé teszi a felhasználók postafiókjába érkező levelek letöltését, de ennél többet nem tud. A beérkező *(és főleg, kiküldött)* leveleket továbbra is az IIS SMTP-szolgáltatása kezeli. Ha azonban figyelmesen elolvastuk az előző részben az SMTP-tartományokra vonatkozó sorokat, emlékezhetünk, hogy az SMTP-kiszolgáló tudása kimerül abban, hogy az egy-egy adott tartományba érkező leveleket *(a címzett ellenőrzése nélkül)* egy közös mappába, a Drop könyvtárba helyezi el. Eddig szó nem volt felhasználókról, és postafiókokról. A kérdés – a hiányzó láncszem – tehát az, hogy az SMTP által fogadott, és tömegesen gyűjtött levél hogyan kerül be a POP3 által kezelt megfelelő felhasználói postafiókba?

Nos, a válasz erre is a POP3 szolgáltatásban rejlik. A POP3-kiszolgáló telepítése után összekapcsolódik az SMTP-vel, és ezután az nemcsak a beérkező leveleket szortírozza szét szépen a regisztrált felhasználói postafiókok között, de még azt is ellenőrzi, hogy létezik-e egy-egy beérkező levélben szereplő e-mail cím az általa kezelt tartományban. A POP3 tehát nem létezik az SMTP nélkül, az SMTP pedig nem képes a levelek szétválogatására a POP3 segítsége nélkül. Ennek a fura szimbiózisnak lesz még látványos hatása a későbbiekben.

A POP3-kiszolgáló telepítése

A POP3-kiszolgálót a megszokott helyen, a *Control Panel →Add or Remove Programs → Add/Remove Windows Components* dialógusablak segítségével telepíthetjük.

A komponensek listájában E-mail Services néven találjuk, amit ha bekapcsolunk, a POP3-kiszolgáló mellett rögtön telepíti az SMTP-szolgáltatást is *(emlékezzünk, szimbiózis)*. Az E-mail Services alkomponense a POP3 szolgáltatás mellett a *POP3 Service Web Administration* is, amely a kiszolgáló rendszergazdai, felügyeleti weboldalához csatolja a POP3 kezeléséhez szükséges alkomponenseket is. Maga a POP3-kiszolgáló természetesen enélkül is működőképes, hiszen az MMC-be épülő konzolmodul mindenképpen felkerül a rendszerre.



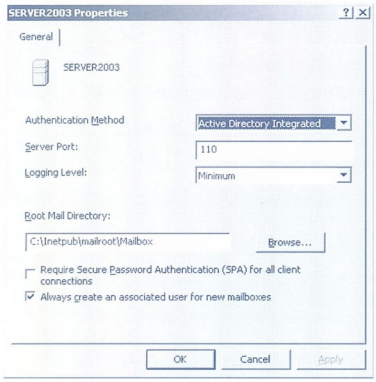
A POP3-kiszolgáló „álneve" E-mail Services

A POP3 felügyeleti MMC konzolja

Ha a szolgáltatás telepítése után a *Start Menü → Administrative Tools → POP3 Service* menüpontra kattintunk, megjelenik a POP3-kiszolgáló felügyeleti eszköze.

Egyszerű szolgáltatáshoz egyszerű konzol

Mielőtt tartományt definiálunk, ellenőrizzük a kiszolgáló általános tulajdonságait a *Server Properties* parancsra kattintva!



A POP3-kiszolgáló tulajdonságlapja. A felhasználóazonosítás módját csak tartomány létrehozása előtt módosíthatjuk!

A dialógusablak legelső kérdése a bejelentkező felhasználók azonosításának módja *(Authentication Method)*. Ezt a beállítást csak akkor módosíthatjuk, ha a POP3-kiszolgálón még nem definiáltunk tartományokat. Ebből következik, hogy egy POP3-kiszolgáló minden egyes tartománya kötelezően ugyanazt az azonosítási módot használja.

Azonosítási módok

A kiválasztható azonosítási módok a következők:

- **Local Windows Accounts:** a felhasználókat a kiszolgáló saját felhasználói adatbázisa tartalmazza, minden postafiókhoz egy helyi felhasználó tartozik, a bejelentkezéshez a felhasználók a helyi felhasználónevüket és jelszavukat használják. Ebben az azonosítási módban a felhasználónévnek tartományok között is egyedinek kell lennie, tehát nem létezhet ugyanazon a kiszolgálón *user1@falatrax.hu* és *user1@falatrax2003.hu* postafiók. A *Local Windows Accounts* azonosítás tartományvezérlőn nem választható.
- **Active Directory Integrated:** a felhasználókat a tartományi adatbázis *(Active Directory)* tartalmazza. Minden postafiókhoz egy tartományi felhasználó tartozik, a bejelentkezéshez a felhasználók a tartományi teljes felhasználónevüket *(például: user1@falatrax.hu)* és jelszavukat használják. Ekkor több tartományban is definiálhatjuk ugyanazt a felhasználónevet *(például user1@falatrax.hu és user1@falatrax2003.hu)* két különböző felhasználóhoz rendelve is. Az *Active Directory Integrated* azonosítás tartományvezérlőn és tartományi tagkiszolgálókon egyaránt választható.
- **Encrypted Password File:** ebben az azonosítási módban a felhasználók jelszaván nem a Windows helyi adatbázisa, vagy az Active Directory, hanem a felhasználók postafiókjában tárolt titkosított fájl tartalmazza. Külön felhasználói fiók létrehozására nincs szükség, a felhasználók a bejelentkezéshez az e-mail címüket *(user1@falatrax.hu)* és a postafiók létrehozásakor megadott jelszavukat használják. Ez az azonosítási mód tartományvezérlőn, tagkiszolgálón és tartományon kívüli kiszolgálón is választható.

A kiszolgáló további beállításai

A dialógusablak következő mezőjében a kiszolgáló portcímét módosíthatjuk *(Server Port:)*, de általában ez nem ajánlott, hiszen az ügyfelek a POP3 szabványos portján *(TCP 110)* keresik a kiszolgálót. A *Logging level* mezőben a naplózás szintjét állíthatjuk be *(None, azaz nincs naplózás, Minimum, Medium és Maximum)*. A bejegyzések az eseménynaplóba kerülnek. A dialógusablak alján látható *Always create an associated user for new mailboxes* mező bekapcsolásának annyi a hatása, hogy új postafiók létrehozásakor a POP3-kiszolgáló felajánlja az új felhasználó létrehozását *(kivéve az Encrypted Password File azonosítási mód esetén, amikor erre nincs szükség)*.

A POP3-kiszolgáló gyökérkönyvtára

A POP3-kiszolgáló által kezelt e-mail tartományok és postafiókok egy könyvtárstruktúra képében öltenek testet. A *Root Mail Directory*: mezőben ennek a struktúrának a gyökérkönyvtárát határozhatjuk meg, ami alapértelmezésben a *C:\Inetpub\mailroot\Mailbox* lesz. A POP3 szolgáltatás ebben a könyvtárban minden tartományhoz egy almappát hoz létre *(a könyvtár neve megegyezik a tartomány nevével)*. A felhasználók postafiókjai a tartományi almappába kerülnek *(P3_usernév.mbx néven)*, felhasználók levelei pedig a postafiók saját mappájába kerülnek. Egy *user1@falatrax.hu* címre küldött levél fájlneve és elérési útja tehát például:

```
C:\Inetpub\mailroot\Mailbox\falatrax.hu\
↳ P3_user1.mbx\P3_20030708113952098200000001.eml
```

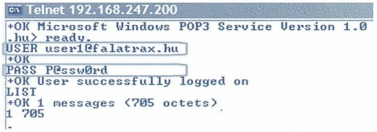
A fenti könyvtárstuktúrához csak az *Administrators* csoport tagjai és a *Network Service (ennek a nevében fut a POP3-kiszolgáló is)* fér hozzá. A gyökérkönyvtár elérési útja módosítható a kiszolgáló tulajdonságlapján, illetve a *winpop* parancs segítségével is, de ha a könyvtárstuktúra már tartalmaz létrehozott tartományokat, postafiókokat és leveleket, a tartalom átviteléről kézzel kell gondoskodnunk. A *Mailbox* könyvtárat és alkönyvtárait *copy* helyett *move* paranccsal kell az új helyre mozgatni, mert a jogosultságok mellett fontos szerepe van az egyes fájlok tulajdonjogának is. A gyökérkönyvtár beállításait a

```
winpop set mailroot <elérési út>
```

paranccsal is módosíthatjuk, ez a parancs ráadásul beállítja a mappák megfelelő biztonsági beállításait is *(a tulajdonjogot nem!)*. A gyökérmappa átállítása után újra kell indítani a *POP3 Service* és az *IIS Admin* rendszerszolgáltatásokat.

Secure Password Authentication

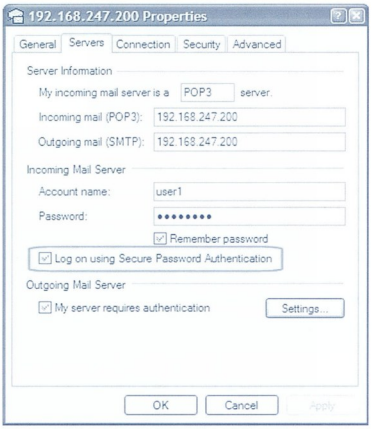
A POP3 az egyik ősprotokoll, ahol a felhasználónév és a jelszó titkosítatlanul utazik a hálózaton.



Titkosítatlan jelszó a POP3 bejelentkezésnél

Ez bizony erőteljes biztonsági rés *(hacsak magát a hálózati forgalmat nem titkosítjuk a kommunikációs csatorna alatt, például IPSec-kel)*. A bejelentkezés biztonságossá tételének másik módja az, ha magát a felhasználóazonosítási protokollt „cseréljük

ki", az úgynevezett *Clear Text*, azaz nyíltjelszavas azonosítás helyett a *Secure Password Authentication (SPA)* metódusra. Az SPA gyakorlatilag integrált Windows authentikációt használ, azaz a bejelentkezéshez a jól ismert NTLM azonosítást használja fel. Ezt a POP3-kiszolgáló tulajdonságlapján található *Require Secure Password Authentication (SPA) for all client connections* opció bekapcsolásával érhetjük el. Az SPA felhasználóazonosítás során a felhasználónév és a jelszó nem utazik titkosítatlanul a hálózaton, a sikeres bejelentkezéshez azonban a levelezőkliensnek is ismernie és használnia kell az SPA-t.



Az SPA bekapcsolása Outlook Expressben

A fenti opció bekapcsolása után a levelezőkliens a szerverhez való legközelebbi csatlakozáskor kérni fogja a felhasználónevet és jelszót. Sajnos *(szerencsére)* ezt információt az Outlook Express biztonsági okokból maradandóan nem tárolja, ezért a program újraindítása után egy alkalommal mindig meg kell majd adni. *(Az ábrán látható dialógusablakban megadott felhasználónevet és jelszót SPA esetén nem használja).* Az igazsághoz hozzátartozik, hogy az Outlook Express, ha SPA-t igénylő kiszolgálóval találkozik, először a felhasználó számítógépén használt felhasználónévvel és jelszóval próbálkozik, és csak akkor kéri a név és jelszó megadását, ha a kiszolgáló az előző bejelentkezési próbálkozást visszautasította.

Az APOP authentikáció

Az SPA erősen Windows-függő protokoll, ezért az Encrypted Password File azonosítási módban működő POP3-kiszolgálóknál az SPA nem használható *(akkor ugyanis nincs Windows felhasználó, akinek a nevét-jelszavát ellenőrizni lehetne).* Ebben az esetben egy másik felhasználóazonosítási protokollt, az úgynevezett APOP azonosítást fogadja el a kiszolgáló. Az APOP leírása az **[RFC1725]**-ben található, és tulajdonképpen pofonegyszerű:

A felhasználó POP3 bejelentkezése APOP-pal

Az APOP azonosítást támogató kiszolgáló beköszönő fejlécében mindig található egy minden egyes csatlakozáskor egyedi azonosító, az ábrán ez például:

```
<19085273@server2003.falatrax2003.hu>
```

Az ügyfél ehhez a fejléchez *(kacsacsőrökkel együtt)* hozzáfűzi a felhasználó titkosítatlan jelszavát, és az egészből egy MD5 hash-t képez. Ezután a POP3 protokollban kiadja az

```
APOP <felhasználó> <MD5-hash>
```

parancsot, és már készen is vagyunk.
A felhasználó jelszava így nem kerül ki a hálózatra, ráadásul az MD5 hash is minden alkalommal más és más, hiszen a kiszolgáló a fejlécben mindig változó adatokat küld *(ez a challenge-response eljárás lényege).*

> *Az Outlook és az Outlook Express a Secure Password Authentication protokollt igen, az APOP parancsot viszont nem támogatja! (A The Bat! viszont például igen, kipróbáltuk, működik.)*

A tartományok létrehozása

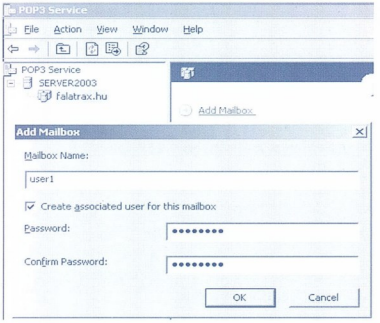
Miután a kiszolgáló tulajdonságait beállítottuk, létrehozhatjuk az első tartományunkat. Ehhez a POP3 konzolban kattintsunk az Add domain parancsra, a megjelenő dialógusablakban pedig adjuk meg a kezelni kívánt tartomány nevét.
Ha a jobb oldali mezőben a tartomány nevére kattintunk, a következő parancsokat adhatjuk ki:

- **New Domain** – új tartomány létrehozása
- **Lock Domain** – a kiválasztott tartomány zárolása *(karbantartási okokból).* Ha egy tartomány zárolt, a bejövő leveleket elfogadja *(a kimenőket pedig természetesen kiküldi, hiszen az az SMTP-szolgáltatás dolga)*, de a felhasználói bejelentkezéseket visszautasítja.
- **Delete Domain** – a kiválasztott tartomány törlése. Vigyázzunk, a tartomány törlése törli az összes postafiókot is, az esetleg bennük található levelekkel együtt!

A POP3 konzolban létrehozott tartomány megjelenik az SMTP-kiszolgálóban is *(szimbiózis)*, Local *(custom)* típusú tartományként, amelynek a *Drop directory* mezője üres. Ez jogos is, hiszen a POP3-tartományokba érkező levelek már nem egy közös Drop könyvtárba, hanem a megfelelő postafiók könyvtárába kerülnek bele *(és ezt az SMTP-kiszolgáló teszi – szimbiózis mégegyszer).*

Postafiók létrehozása

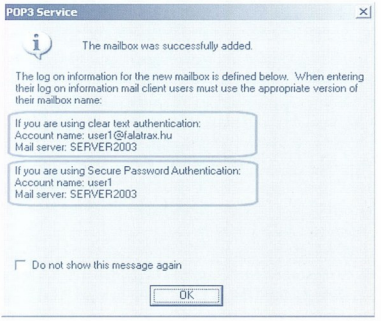
Miután a tartományt létrehoztuk, következhetnek a postafiókok. Kattintsunk a POP3 konzol bal oldali fájában a kiválasztott tartomány nevére, majd pedig a megjelenő *Add Mailbox...* feliratra!



POP3 postaláda létrehozása

Ha bekattintjuk a *Create associated user for this mailbox* mezőt, és megadunk egy *(pontosabban két)* jelszót, a varázsló megkíséri létrehozni a felhasználót *(a kiszolgáló beállításától függően a helyi Windows adatbázisban, vagy az Active Directory-ban)*. Emlékezzünk azonban arra, hogy *Local Windows Accounts* azonosítás esetén nem lehet két azonos nevű postafiókunk, még különböző tartományokban sem. Az *Active Directory Integrated* azonosítás kiválasztásakor ez a korlátozás nem érvényes, ekkor generálhatunk azonos nevű postafiókokat, ugyanis a felhasználók vagy a teljes felhasználónevükkel *(user1@falatrax.hu formátumban)*, vagy pedig a *Pre-Windows 2000 Logon Name* nevüket használva jelentkeznek be. *(A másodiknak létrehozott postafiókhoz generált felhasználói fiók Pre-Windows 2000 logon neve módosulni fog – a 000 sorszámot illeszti hozzá a rendszer – így elkerülhető lesz a névütközés. Tipp: ha van olyan nevű e-mail tartományunk, ami megegyezik a Windows-tartomány nevével, először ebben a tartományban hozzuk létre a felhasználót, és csak később a többiben, különben „The specified user already exists." hibaüzenetet kapunk.)* *Encrypted Password File* azonosítás esetén a jelszó megadása mindig kötelező, de a postaládához felhasználó nem jön létre.

Akárhogy is tettünk, a postafiók létrehozásának eredménye egy dialógusablak, amely tartalmazza a felhasználó bejelentkezéséhez szükséges felhasználónevet *(adott esetben lehet, hogy két különbözőt, a nyílt jelszavas és az SPA azonosítás esetére)*:



A létrehozott postafiók eléréséhez szükséges felhasználónevek

A néhány sorral ezelőtt vázolt esetben előfordulhat, hogy a nyíltjelszavas és SPA azonosításhoz használt felhasználónév teljesen eltér *(például: user1@falatrax.hu, ugyanakkor SPA-hoz: user1000)*. Ezért a postafiók létrehozásakor mindig figyelmesen olvassuk el, jegyezzük fel, és közöljük a felhasználóval a megjelenített adatokat.

A postafiók parancsai

Ha a POP3 konzolban kiválasztunk egy postafiókot, az alábbi műveletek közül választhatunk:

- **Add Mailbox** – új postafiók létrehozása
- **Lock Mailbox** – postafiók zárolása. A zárolt postafiókba küldött levelek megérkeznek, de a postafiókba történő bejelentkezést és a levelek letöltését a kiszolgáló letiltja.
- **Delete Mailbox** – a postafiók törlése *(vigyázat, ez törli a felhasználó összes levelét is!)*

Felhasználók és csoportok

Amikor Windows-, vagy Active Directory-beli felhasználót hozunk létre a varázsló segítségével, a felhasználó tulajdonságlapjának *General* oldalán a varázsló kitölti a felhasználóhoz tartozó e-mail címet is. Mindig ellenőrizzük azonban, hogy ez sikerült-e neki, különben kellemetlen meglepetések érhetnek bennünket.

Amikor a POP3-kiszolgáló nem tartományvezérlőn fut, a varázsló a létrehozott felhasználókat beleteszi a helyi *POP3 Users* csoportba is. Ez a csoport lehetővé teszi, hogy a felhasználók elérhessék a POP3-kiszolgálót, ugyanakkor egyéb módon ne legyen joguk bejelentkezni a kiszolgálóra. Tartományvezérlők esetén ennek a csoportnak nincs szerepe, hiszen oda a tartományi felhasználók egyébként sem léphetnek be.

A felhasználó jelszavát több módon is megváltoztathatjuk: egyrészt, *Local Windows Accounts* és *Active Directory* Integrated azonosítás esetén a Windowsban már megszokott módon, másrészt pedig, bármikor a *winpop* paranccsal:

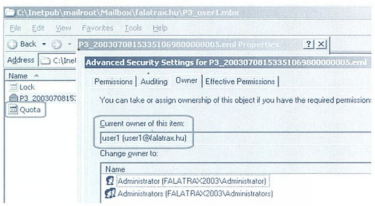
```
winpop changepwd <user@domain> <új jelszó>
```

Kvóták és korlátozások

A felhasználóknak küldött terjedelmes e-mailek hamar megtöltik a rendelkezésükre álló lemezterületet. A postafiók méretét ezért a Windows beépített kvótarendszerének segítségével korlátozhatjuk *(amennyiben a Mailbox gyökérkönyvtár NTFS partíción található, hiszen kvóta is csak ott létezik)*. Egy NTFS partíción minden egyes fájlnak és mappának van tulajdonosa *(ez általában az a felhasználó, aki létrehozta a fájlt)*. A kvótarendszer pedig a fájlok tulajdonjoga alapján számolja és összegzi az egyes felhasználók által használt lemezterületet, és ha a használt terület mérete eléri az előre beállított határt, a Windows a lemezre írást megakadályozza *(kivéve az Administrators csoport tagságát, akikre nem vonatkoznak a kvóta korlátozásai)*.

A beérkező leveleket azonban nem a felhasználó, hanem maga a szolgáltatás helyezi el a postafiókokban, így elvileg a levelek tulajdonosa is a *Network Service* lenne – ha nem módosítaná a fájlok tulajdonosának beállításait a megfelelő módon. Ehhez a felhasználó belső biztonsági azonosítójára *(Security Identifier, SID)* van szükség, amelyet a felhasználó postaládájában, a *Quota* nevű fájlban tárol a rendszer. Amikor egy levél érkezik, a kiszolgáló a megfelelő felhasználó postaládájában található *Quota* fájlból kiolvassa a felhasználó SID-jét, és beállítja a létrehozott fájlon.

A *Quota* fájl *Local Windows Accounts* és *Active Directory Integrated* azonosítás esetén a postafiók létrehozásakor automatikusan létrejön. Az *Encrypted Password File* azonosítás használatakor azonban nincs felhasználó, így nincs kvótafájl sem.



A postafiókban tárolt levelek tulajdonosa maga a felhasználó – így érvényesülhet rá a kvóta is

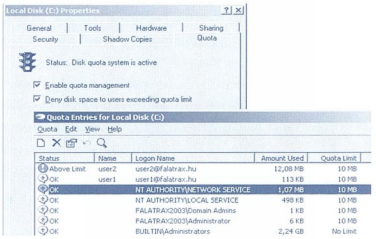
Ha mégis szeretnénk kihasználni a kvótarendszer előnyeit, magunknak kell létrehozni a felhasználókat, és elkészíteni a postafiókokban a megfelelő *Quota* fájlt. Ehhez a *winpop* parancsot használhatjuk:

```
winpop createquotafile <postaláda>
↳/user:<felhasználó>
```

például:

```
winpop createquotafile user1@falatrax.hu
↳ /user:administrator
```

A parancs futtatása után minden, az adott postafiókba beérkező levél a felhasználó kvótáját terheli. *(Ne keverjük össze a postafiók „felhasználóját" a kvótázáshoz használt felhasználóval. A fenti művelet után az Administrator felhasználónak továbbra sem lesz joga bejelentkezni a user1@falatrax.hu postafiókba.)*
A kvótázást azon a meghajtón kell bekapcsolni, amin a postafiókok gyökérkönyvtára szerepel *(éppen ezért érdemes a postafiókoknak egy külön partíciót fenntartani)*. A meghajtó tulajdonságlapjának Quota oldalán válasszuk az *Enable quota management* és a *Deny disk space to users exceeding quota limit* opciókat, majd állítsuk be az alapértelmezett kvótaméretet. Az *Apply* gombra kattintva a kvótarendszer bekapcsol, és elkezdi felépíteni a kvótaadatbázist *(a közlekedési lámpa ekkor sárgán világít)*. Amikor zöldre vált, a kvótaadatbázis elkészült, és a *Quota Entries…* gombra kattintva meg is tekinthetjük. Az egyes bejegyzéseken jobb gombbal kattintva felhasználónként is módosíthatjuk a kvótabeállításokat. A *Status* oszlopban láthatjuk, hogy az egyes felhasználók milyen viszonyban vannak a beállított határokkal.



Ha engedélyeztük a kvótázást, a kvótabejegyzéseknél jól látszik a postafiókok által felhasznált terület

Ha egy postafiók megtelt, a beérkező levelet a kiszolgáló nem kézbesíti, erről a tényről pedig *Non-Delivery Report (NDR)* üzenetet küld a feladónak.

Tartományi kvótázás

A most leírtak alapján a kvóta minden egyes felhasználóra külön-külön érvényesül. Azonban megtehetjük, hogy ugyanazt a Windows-felhasználót egynél több, sőt, akár teljes tartománnyi felhasználóhoz is hozzárendelhetjük a *winpop* segítségével *(vagy, ha a kiválasztott felhasználó postafiókjából a Quota fájlt kézzel bemásoljuk a többi felhasználó postafiókjába)*. Ekkor a kvóta egyidejűleg mindenkire érvényesül majd, hiszen mindenkinek ugyanazzal a SID-del jönnek létre a beérkező leveleket tartalmazó fájlok.

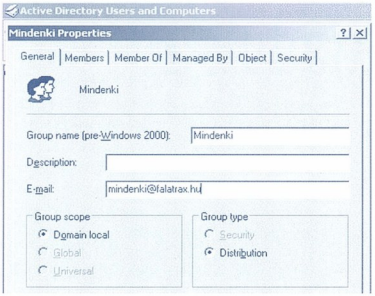
Áttérés a felhasználóazonosítási üzemmódok között

A felhasználóazonosítási üzemmódok között két útvonalon „biztosít" átjárást a POP3-kiszolgáló. Az első, ha *Local Windows Accounts* azonosítással futó POP3-kiszolgáló „alatt" tartományvezérlőt telepítünk a számítógépre. Ekkor – definíció szerint – a *Local Windows Accounts* azonosítási mód megszűnik. Az áttérésnek az a sajnálatos útja, hogy a tartományvezérlő telepítése előtt törölnünk kell a POP3-kiszolgáló minden postafiókját és a definiált tartományokat, majd a tartományvezérlő telepítése és az azonosítási mód átállítása után újra létre kell hozni azokat.
A másik átjárás barátságosabb: ennek során az *Encrypted Password File* azonosítási módot váltja fel az *Active Directory Integrated* azonosítás. Ekkor – bár a postafiókokat az üzemmódváltás miatt itt is létre kell majd hozni – a meglévő POP3 felhasználókat jelszavastul „be lehet emelni" a címtárba, a következő parancs segítségével:

```
winpop migrateToAD <user@domain>
```

SMTP LDAP Routing

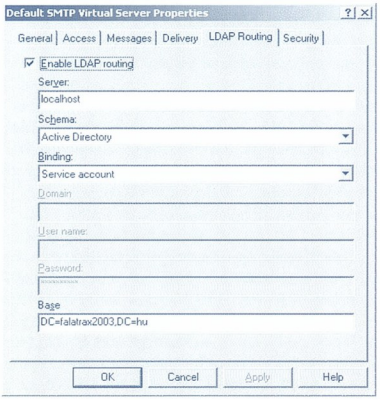
Az előző számban az SMTP-kiszolgálóról volt szó, de a SMTP-szolgáltatás egy opcióját tudatosan későbbre hagytuk. Ez az *LDAP Routing* néven ismert funkció, ami a levelek postázásához képes felhasználni a különféle *(Active Directory, Exchange, stb.)* címtárakban tárolt információkat. Egészen pontosan e-mail terjesztési listákról *(Distribution List)* van szó, ami saját e-mail címmel rendelkezik, és minden erre a címre küldött levelet minden csoporttag-felhasználó megkap. Nyilvánvaló hogy ez az SMTP-kiszolgáló feladata, de addig, míg a postafiókok kezelése *(hála a POP3-nak)* nem volt megoldva, ennek a funkciónak nem volt sok értelme. No de most!
Tegyük fel, hogy adott egy e-mail tartomány, Active Directory alapú felhasználóazonosítással. Benne három felhasználó, név-(emailcím-) szerint *administrator@falatrax.hu, user1@falatrax.hu, user2@falatrax.hu*. Egy szép napon valaki a falatrax.hu teljes ügyfélkörének szeretne levelet küldeni, mondjuk úgy, hogy a címzetthez *mindenki@falatrax.hu*-t ír, a többit pedig rábízza a kiszolgálóra. Az eredmény – hiszen a fenti postafiók nem létezik – egy NDR lesz, miszerint a *mindenki@falatrax.hu* címre nem kézbesíthető levél.
Hozzunk most létre az Active Directory-ban egy terjesztési csoportot *(Distribution Group)*, és tagjainak vegyük fel a fenti három felhasználót!



Terjesztési csoport az Active Directory-ban

Ne felejtsük el beállítani a csoport e-mail címét! Ezután indítsuk el az *Internet Information Services Manager* MMC konzolt, majd nyissuk meg a *Default SMTP Virtual Server* tulajdonságlapját! Ennek *LDAP Routing* oldalán lehet engedélyezni a címtárhoz való kapcsolódást. A beállítások a következők:

- **Enable LDAP Routing** – az LDAP Routing engedélyezése
- **Server** – a címtárkiszolgáló neve
- **Schema** – a címtár típusa, lehet *Active Directory, Site Server Membership Directory és Exchange LDAP Service*
- **Binding** – a kapcsolódáshoz használt bejelentkezési adatok. Lehet *Anonymous, Service account (ekkor a Network Service szolgáltatás nevében kapcsolódik), plain text (nyílt szöveges)*, és *Windows SSPI*. Utóbbi két esetben meg kell adnunk a használni kívánt tartományt, felhasználónevet és jelszót.
- **Base** – a címtárbeli keresés kiindulópontja. Lehet például a tartomány neve, de ha az objektumainkat a címtár egy jól definiált pontjára *(például szervezeti egységbe)* tömörítettük, itt megadhatjuk azt is. Ez felgyorsítja a keresést.



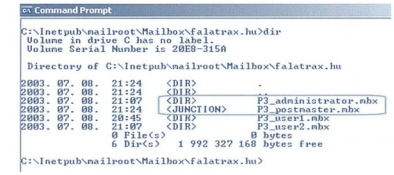
Az LDAP kapcsolódási pont beállításai

Az *IIS Admin* szolgáltatás újraindítása után az SMTP-kiszolgáló már ellenőrzi a címeket a címtárban, elfogadja és kifejti *(kézbesíti)* a terjesztési listák címére *(például a mindenki@falatrax.hu címre)* érkező leveleket is.

Egy trükk a végére: aliasok

„Hivatalosan" nincs mód arra, hogy egy postafiókhoz több e-mail címet rendeljünk, holott ez az általános gyakorlatban bevett szokás. Mégis van megoldás, ami az NTFS fájlrendszer egy ritkán használt funkciójára, a *hard link*-re alapul. Mint tudjuk, a szolgáltatás a beérkező leveleket a könyvtárstruktúra segítségével szortírozza, helyezi el a megfelelő postafiókba. Az NTFS hard link-nek köszönhetően megtehetjük, hogy olyan elágazási pontot *(junction point)* hozunk létre, ami más néven, de egy adott könyvtárra mutat. Az ötlet egyszerű: ha egy felhasználó postafiókja a *P3_usernév.mbx* könyvtárban található, hozzunk létre egy *P3_másiknév.mbx* mappát, ami fizikailag ugyanerre a könyvtárra mutat. Ehhez a [**rktools2003**] címről letölthető Windows 2003 Resource Kit Tools egyik segédprogramját, a *linkd.exe*-t használhatjuk. Tegyük fel például, hogy az *administrator@falatrax.hu* postafiókhoz szeretnénk felvenni mondjuk a *postmaster@falatrax.hu* címet is. Ehhez először hozzuk létre a szokásos módon az administrator postafiókját, majd menjünk be a *falatrax.hu* könyvtárba és adjuk ki a következő parancsot:

```
linkd P3_postmaster.mbx P3_administrator.mbx
```



A hard linket felismerhetjük a <JUNCTION> feliratról.

Ekkor látszólag létrejön egy P3_postmaster.mbx mappa, ami valójában fizikailag a P3_administrator.mbx könyvtárra mutat. Ennek köszönhetően a *postmaster@falatrax.hu* címre beérkező levelek az Administrator postafiókjába potyognak majd be. A leveleket letölteni, bejelentkezni természetesen változatlanul, az Administrator felhasználóval kell és lehetséges. Ha törölni szeretnénk a felhasználót, előbb töröljük a linket, különben az a semmibe fog mutatni:

```
linkd P3_postmaster.mbx /d
```

Fülöp Miklós
mick@inetcom.hu

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

Tanúsítványkiadók a Windowsban

Mi van egy tanúsítványban?

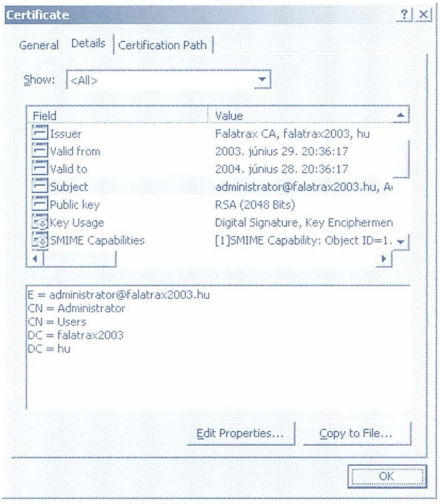
Cikkünk következő részében először az X.509 digitális tanúsítványok mélyreható boncolgatására kerül sor, majd visszatérünk a Windows Server 2003, Enterprise Edition tanúsítványkiadójának újdonságaira.

A tanúsítványok mezői

A PKI tanúsítványok digitálisan aláírt adatcsomagok, amelyek a tulajdonos publikus kulcsán és a tulajdonos adatain kívül sok más információt is tartalmaznak. Ezeket az információkat a tanúsítványt felhasználó és kezelő alkalmazások *(beleértve magát a Windows operációs rendszert)* kezelik és dolgozzák fel.

Ha egy tanúsítványt megnyitunk *(kettőt kattintunk rá)*, a Certificate dialógusablak jelenik meg. A dialógusablak első *(General)* oldalán a tanúsítvány legfontosabb jellemzőit *(tulajdonos, kiadó, érvényességi idő, privát kulcs megléte – vigyázat ez nem azt jelenti hogy a tanúsítvány tartalmazná a privát kulcsot!)* sorolja fel a rendszer. A részletes adatok a Details oldalon találhatók.

A tanúsítványok felépítését *(és mezőit)* az **[RFC2459]** szabvány írja le bitekig történő részletességgel. Ezeket a mezőket mutatjuk be a következő oldalakon.



Az X.509 tanúsítvány mezői

A tanúsítványok mezői három csoportba sorolhatók. A mezők általános információk *(ezek megléte minden tanúsítványban kötelező)*, bővítmények, és kritikus bővítmények lehetnek. A bővítményeket zöld nyíl, a kritikus bővítményeket sárga, felkiáltójelet tartalmazó háromszög jelzi. Ha egy alkalmazás egy tanúsítványban kritikus bővítményt „vár", de az nincs jelen, vagy a tanúsítvány olyan kritikus bővítményt tartalmaz, amit a Windows nem ismer fel, a tanúsítvány érvénytelennek számít. Lássuk először az általános mezőket *(Version 1 Fields)* sorban:

- **Version:** a tanúsítvány X.509 verziója. Az RFC 2459 és a Windows a v3-as tanúsítványverziót használja.
- **Serial Number:** a tanúsítvány sorozatszáma. CA-n belül egyedi, minden kiadott tanúsítványt egyértelműen azonosít. Ez az azonosító szerepel a visszavont tanúsítványok listájában is.
- **Signature Algorithm:** a tanúsítvány digitális aláírásához használt algoritmus
- **Issuer:** a tanúsítványt kiadó szervezet *(CA)* neve *(X.501 formátumban)*
- **Valid from:, Valid to:** a tanúsítvány érvényességi ideje *(tól-ig)*.
- **Subject:** a tanúsítvány *(publikus kulcs)* tulajdonosának neve *(X.500 formátumban)*. Ez a mező lehet üres, ha a tulajdonos nem CA, és ha a tanúsítvány tartalmazza a Subject Alternative Name mezőt. *(A tanúsítványok tulajdonosának ellenőrzésekor a rendszer mindig mindkét jellemző értékét ellenőrzi.)*
- **Public Key:** a publikus kulcsot és a titkosítási algoritmus azonosítóját tartalmazó mező *(tehát nem bitről-bitre a publikus kulcs!)*

Eddig a kötelező *(v1 verziójú tanúsítványokban definiált)* mezők. A következő lista a bővítménymezőket tartalmazza. Ezek a bővítmények csak v3 verziójú tanúsítványban szerepelhetnek, és jelenlétük sem mindig kötelező.

- **Subject Alternative Name:** a tulajdonos egyéb nevei. Ebben a mezőben a tanúsítvány tulajdonosának egyéb azonosítói találhatók. Egy azonosító lehet e-mail cím *(illetve User Principal Name, a felhasználó Windows 2000 tartomány-beli, „e-mailcímszerű" neve)*, IP-cím, DNS-név és Uniform Resource Identifier *(URI)*. Minden esetben, amikor bárki a tanúsítvány tulajdonosára vonatkozó adatokat ellenőriz, vagy dolgoz fel, a Subject mezőben található X.500-formátumú név mellett kezelnie kell a Subject Alternative Name mező tartalmát is.
- **Key Usage:** a tanúsítványban található kulcs felhasználási területei. Az alábbiak kombinációja lehet *(ha nincs*

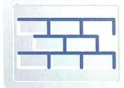
megadva, a felhasználási területeket nem korlátozzuk): Digital Signature *(digitális aláírás)*, Non Repudiation *(digitális aláírás – letagadhatatlanság biztosítása)*, Key Encipherment *(kulcstitkosítás)*, Data Encipherment *(adattitkosítás)*, Key Agreement *(kulcsmenedzsment – pl. Diffie-Hellmann protokollhoz)*, Certificate Signing *(tanúsítványok digitális aláírása – csak CA tanúsítványokban lehet jelen)*, CRL Signing *(tanúsítvány-visszavonási lista digitális aláírása)*, Encipher Only *(csak titkosítás)*, Decipher Only *(csak dekódolás)*

- **SMIME Capabilities:** titkosítási algoritmusok azonosítói *(OID-k)*, amelyekhez a tanúsítvány *(és a benne tárolt kulcs)* felhasználható. Egy Windows 2003 CA által kiadott tanúsítvány az alábbi OID-ket tartalmazza: 1.2.840.113549.3.2 *(RC2-CBC)*, 1.2.840.113549.3.4 *(RC4)*, 1.3.14.3.2.7 *(DES-CBC)*, 1.2.840.113549.3.7 *(DES-EDE3-CBC)*. Ezt a mezőt az S/MIME v2 specifikációjában **[RFC2311]** találjuk meg először. Az alkalmazások egyelőre nem várják el a mező meglétét *(a Windows 2000 CA-ja által kiadott tanúsítványokban ez a mező például még nem is szerepel)*.

Object Identifier *(OID)*: Azonosítórendszer, amelyet világszerte különböző célokra használnak. Az adatbázist az ITU, az IANA, az ANSI és más regisztrátorok kezelik. Az azonosítórendszer hierarchikus, a hierarchia szintjeit az azonosítóban található pontok különítik el. *(Az Internet OID-je például 1.3.6.1, a titkosítási algoritmusoké például 1.2.840.113549.3)*. Saját OID-t cégek is kaphatnak, azokat tovább „bontva" *(azaz bővítve)* bárki saját OID-ket is generálhat. A Microsoft egyik OID-je például 1.3.6.1.4.1.311 – minden OID, ami így kezdődik, a Microsoft „sajátja". A [KB287547]-es tudásbáziscikkben sok microsoftos, kriptográfiai területen használt, „belső" OID leírását megtaláljuk.

- Subject Key Identifier: A tanúsítványban található publikus kulcs egyedi azonosítója *(általában a publikus kulcsból készült SHA-1 hash)*. Ezt az adatot általában a tanúsítványlánc felépítésében használjuk *(az Authority Key Identifier mezővel együtt)*.
- Authority Key Identifier: A tanúsítvány digitális aláírásához használt privát kulcs publikus párjának egyedi azonosítója *(azaz, a tanúsítványt kiadó CA publikus kulcsából készült SHA-1 hash)*. Egy tanúsítványban található AKI érték megegyezik a kiadó CA tanúsítványában található SKI mező értékével.
- Certificate Template Name/Information: a tanúsítvány kiadásához használt tanúsítványsablon neve *(Windows 2003 CA esetén verziószáma is)*. Ez a mező nem szerepel az **[RFC2459]** specifikációban.
- CRL Distribution Points: a tanúsítványt kiadó CA által közzétett tanúsítványvisszavonási listák *(CRL-ek)* elérési útja(i). Értéke egy vagy több URI *(Uniform Resource Identifier)*, amelyek lehetnek ldap:// *(X.509 címtárbeli)*, http://, ftp://, file:// *(fájlrendszerben közvetlenül elérhető)* címek. Az alkalmazások az itt található információ segítségével tölthetik le a CRL-t, és ellenőrizhetik a tanúsítvány visszavonási státuszát. A mező megléte nem kötelező, de ajánlott.

Magyar viszonyok: A cikk írásának pillanatában *(2003. június 29.)* két, HIF által minősített tanúsítványkiadó szolgáltatást nyújtó céget, a NetLock Kft-t **[NetLock]**, és a MÁV Informatika Kft-t **[MAVCA]** „teszteltük".



A NetLock által kiadott teszttanúsítványok nem tartalmazzák a CRL Distribution Point mezőt. *(Rákérdeztünk: ennek oka, hogy elég sok alkalmazás egyszerűen lefagy, ha nem éri el a CDP-t – márpedig belső, védett hálózatról ennek nem is kicsi az esélye. A NetLock ezért átmeneti megoldásként – amíg az alkalmazások „ki nem javulnak" – egyrészt lehetővé teszi a **[NetLockCRL]** címről az aktuális CRL-ek letöltését, amelyek importálhatók a rendszerbe, másrészt a tanúsítvány kérhető kitöltött CDP-vel is.)*

A MÁV Informatika CA-jától kapott teszttanúsítvány teljes és érvényes CRL hivatkozást tartalmazott.

- **Enhanced Key Usage:** Kiterjesztett kulcshasználati információk. Kriptográfiai alkalmazások OID-it találjuk itt, amelyek esetén a tanúsítvány *(és a benne található kulcs)* alkalmazható. Ez a mező nem szerepel az **[RFC2459]**-es specifikációban, viszont létfontosságú a Windows 2000-alapú rendszerekben. A Windows 2003 CA már képes az Application Policies mező kezelésére is, amely ugyanezeket a feladatokat látja el *(és szerepel is az **[RFC2459]**-ben)* és idővel teljesen felváltja majd az EKU-t..

  Ebben a mezőben olvasható OID-k határozzák meg tehát azt, hogy egy tanúsítvány mire használható. Mielőtt egy alkalmazás felhasználna egy tanúsítványt, ellenőrzi ennek a mezőnek a tartalmát. Néhány példa: Client Authentication *(Ügyfélazonosítás, 1.3.6.1.5.5.7.3.2)*, Secure Email *(Biztonságos levelezés, 1.3.6.1.5.5.7.3.4)*, Smart Card Logon *(Bejelentkezés Smart Card-dal, 1.3.6.1.4.1.311.20.2.2)*, Encrypting File System *(1.3.6.1.4.1.311.10.3.4)*, File Recovery *(EFS fájlok helyreállítása, 1.3.6.1.4.1.311.10.3.4.1)*, Server Authentication *(Kiszolgálóazonosítás, 1.3.6.1.5.5.7.3.1)*, stb. Ha nincs EKU mező, a tanúsítvány bármilyen területen felhasználható, amit a Key Usage mező engedélyez.

- **Application Policies:** ugyanarra a célra való, mint a fent bemutatott EKU mező. A Windows Server 2003, Enterprise Edition CA szolgáltatása által kiadott tanúsítványok már tartalmazhatják ezt a mezőt. A Windows Server 2003 kompatibilitási okok miatt támogatja az EKU használatát is, de ha az AP mező megtalálható, az EKU mező tartalma irreveláns *(a kiadott tanúsítványokban egyébként az EKU és az AP mező tartalma megegyezik – ha van)*. A meglévőek mellett saját Application Policy-t is létrehozhatunk *(saját OID-vel)*.
- **Certificate Policies:** más néven Issuance Policies: a tanúsítványok kiadásával kapcsolatos eljárás „biztonsági szintjét" meghatározó érték *(csak Windows Server 2003 alatt érvényes)*. Ez biztonsági szint *(illetve a hozzájuk tartozó eljárások)* relatív fogalmak – jelentésük a felhasználási környezettől függően más és más lehet. A Microsoft négy szintet definiál: ezek közül három a Low Assurance *(1.3.6.1.4.1.311.21.8.x.y.z.1.400)* – alacsony, Medium Assurance *(1.3.6.1.4.1.311.21.8.x.y.z.1.401)* – közepes, High Assurance *(1.3.6.1.4.1.311.21.8.x.y.z.1.402)* magas

biztonsági szint. Az x,y,z érték minden Active Directory erdőben egyedileg, véletlenszerűen generált érték. Ezzel is biztosítható, hogy az egyes szintekhez hozzárendelt eljárási rend „helyi" jellegű, más rendszerekkel automatikusan össze nem egyeztethető. *(Az ilyen párosításra egyébként van mód, de erről a következő részben lesz szó.)* Az előre definiált szintek mellett saját értékeket is létrehozhatunk *(ismét csak saját OID-vel azonosítva).*

A biztonsági szint kikényszerítése is csak egyetlen ponton van kidolgozva „gyárilag": minden CA csak olyan tanúsítványokat adhat ki, amelyekre a CA saját tanúsítványában található Certificate Policies mező tartalma feljogosítja. Ha a CA tanúsítványában ez a mező nem szerepel, nincs korlátozás; ugyanezt jelenti, ha a CA tanúsítványában az előre definiált negyedik, All Issuance *(2.5.29.32.0)* érték szerepel.

- **Basic Constraints:** a mező értéke határozza meg, hogy a tanúsítvány végfelhasználó, vagy tanúsítványkiadó tanúsítványa-e. *(A Windows is ebből dönti el, hogy egy importált tanúsítványt automatikusan a tanúsítványtár Personal vagy a Trusted illetve Intermediate Certification Authorities mappájába helyezi el).* Ha a Basic Constraints mezőben a Subject Type jellemző értéke CA, tanúsítványkiadói tanúsítvánnyal van dolgunk. A mező másik, Path Length Constraint értéke azt határozza meg, hogy a tanúsítvány tulajdonosa *(ha CA)*, adhat-e ki gyermek CA-k részére tanúsítványt, vagy sem. Erre is visszatérünk még a következő részben, egyelőre legyen elég annyi, hogy a jellemző 0 értéke azt eredményezi, hogy a CA csak végfelhasználókat szolgálhat ki. Ha ez a mező egy tanúsítványban megtalálható, mindig kritikus.

Ezzel végére értünk a „hétköznapi" tanúsítványok mezői bemutatásának. Van még néhány fontos mező, ami tanúsítványkiadók tanúsítványai esetén lesz majd érdekes – a következő részben. Addig is, két mező maradt még ismeretlen, amelyek bár az **[RFC2459]**-ben nem szerepelnek, mégis sokszor találkozunk velük, ez pedig a tanúsítvány „ujjlenyomata".

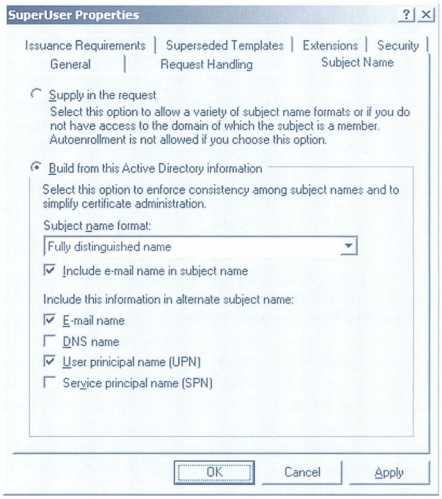
- Thumbprint Algorithm: a tanúsítvány egyedi azonosítójának *(„ujjlenyomatának")* elkészítéséhez használt algoritmus azonosítója
- Thumbprint: a tanúsítvány egyedi azonosítója *(ennek alapján bármely tanúsítvány egyértelműen azonosítható – ne keverjük össze a sorozatszámmal, ami csak CA-n belül egyedi!)*

A v2-es tanúsítványsablonok jellemzői

Az előző részben elkezdtük a v2-es *(azaz írható és módosítható)* tanúsítványsablonok beállításainak bemutatását. Az Active Directory-beli sablonok tulajdonságlapjának Request Handling oldalán találunk egy beállítást, amely a Do the following when the subject is enrolled... névre hallgat. Az opciók a privát kulcs kezelését, illetve az automatikus tanúsítványkérés működését hangolják:

- Enroll subject without requiring any user input: tanúsítvány kiadása felhasználói beavatkozás nélkül
- Prompt the user during enrollment: a tanúsítvány kérése előtt a felhasználót kis felugró buborékablakban értesítjük, hogy automatikus tanúsítványkérés kezdődik
- Prompt... and require user input when the private key is used: automatikus tanúsítványkérés esetén értesíti a felhasználót és bekapcsolja a privát kulcs megerősített védelmét

A Subject Name oldalon a tulajdonos nevével *(most már tudjuk, hogy a tanúsítvány Subject és Subject Alternative Name mezőjével)* kapcsolatos opciókat találjuk:

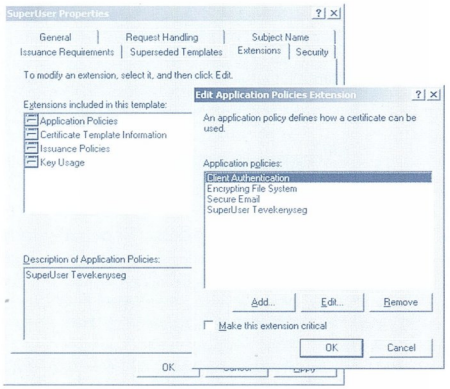


A tanúsítvány tulajdonosának adatai

A Supply in the request opció esetén minden adatot a tanúsítványkérésben küldenek el nekünk. Ami számunkra igazán érdekes, az a Build from this Active Directory information – azaz, amikor az adatok a címtárból származnak. Az első beállítási lehetőség a Subject name format. Itt meghatározhatjuk, hogy a tanúsítvány Subject mezőjébe a tulajdonos teljes, Active Directory-beli LDAP neve *(pl. CN=Administrator, CN=Users, DC=falatrax2003, DC=hu – azaz a Fully distinguished name)*, a rövid neve *(CN=Administrator – azaz a Common Name)*, vagy semmi *(None)* sem kerüljön. *(Gondoljunk arra, hogy ezt a szabvány megengedi, ha a Subject Alternative Name mező ki van töltve).* Ha akarjuk, itt meghatározhatjuk azt is, hogy a felhasználó e-mail címe bekerüljön-e a Subject mezőbe.

A következő opciókkal azt határozhatjuk meg, hogy mi kerüljön a Subject Alternative Name mezőbe: e-mail cím, DNS-név, User Principal Name *(UPN)* vagy Service Principal Name *(SPN)*. Vigyázzunk, hogy e-mail címe csak felhasználónak, DNS-neve pedig csak számítógépnek van. Ha a tanúsítványsablonban ezeket bekapcsoljuk, és a tulajdonos nem rendelkezik valamelyik adattal *(az e-mail cím a felhasználó Active Directory-beli objektumának General oldalán található E-mail mező tartalma)*, a Policy Agent visszautasítja a ta-

núsítvány kiadását. Erről a Certificate Authority konzol Failed Request listájából is értesülhetünk.
A tanúsítványsablon tulajdonságlapjának Superseded Templates oldalán a tanúsítványsablon által érvénytelenített más tanúsítványsablonok találhatók. Az itt felsorolt tanúsítványok helyett a CA már az új típusú tanúsítványt fogja kiadni. Az Extensions oldalon a tanúsítványbővítmények beállításait látjuk.

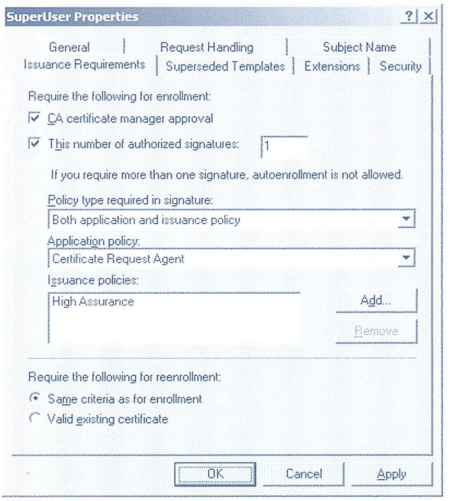


A tanúsítványsablon bővítményeinek beállításai

Itt beállíthatjuk, hogy a sablonból készült tanúsítvány milyen Key Usage, Application Policies illetve Issuance Policies bővítményeket tartalmazzon. Ha az Issuance és Application Policies sor kiválasztása után az Edit... gombra kattintunk, megjelenik a bővítmény saját ablaka, ahova felvehetjük az előre definiált bejegyzéseket *(jusson eszünkbe, hogy az Application Policies bővítmény tartalmával határozzuk meg a tanúsítvány Extended Key Usage mezőjének értékét is)*. Ha az Add... gombra kattintunk, kiválaszthatjuk a listához hozzáadni kívánt OID-ket, sőt a megjelenő ablak New... gombjára kattintva új elemeket is hozhatunk létre. Ehhez névre és OID-re lesz szükségünk, de az OID mezőben szerencsére a Windows automatikusan felkínál egy garantáltan egyedi *(az Issuance Policy leírásánál bemutatott módon generált)*, véletlenszerű OID-t, amit használhatunk, ha akarunk. Az ábrán látható sablonba így kerülhetett bele a SuperUser Tevékenység feliratú Application Policy elem.

Kiadási feltételek

Egy oldal maradt hátra a tanúsítványsablonok jellemzői közül: az Issuance Requirements, azaz kiadási feltételek.
A következő ábrán egy jócskán megszigorított kiadási feltételekkel rendelkező tanúsítványsablont látunk. A CA certificate manager approval opcióról az előző részben már elmondtuk, hogy bekapcsolása esetén a sablonból készült tanúsítványt az Enterprise CA-k sem adják ki automatikusan, hanem minden esetben megvárják egy rendszergazda hozzájárulását *(Certificate Authority MMC konzol, Pending Requests lista, Issue parancs)*.
Emellett azonban létezik egy még komolyabb eljárás is: ha akarjuk, a tanúsítvány kiadását digitálisan speciálisan aláírt tanúsítványkéréshez is köthetjük. Normális esetben egy tanúsítványkérést csak a tanúsítvány leendő boldog tulajdonosa ír alá, de mi kikényszeríthetjük, hogy a tanúsítványkérést elküldés előtt megadott típusú tanúsítvánnyal *(tanúsítványokkal)* rendelkező ember*(ek)* is aláírják.



A sablonból készült tanúsítvány kiadásának szigorú feltételei

Ha a This number of authorized signatures: opciót bekapcsoljuk, a webes tanúsítványkérés már csak fájlba menteni lesz hajlandó minden kérést *(jól is teszi, különben a kérést a Policy Modul a megfelelő aláírás híján azonnal vissza is utasítaná)*. Az opció utáni mezőben meghatározhatjuk, hogy a tanúsítványkérést elküldés előtt mennyi *(természetesen egymástól különböző)* digitális aláírással kell ellátni *(az alapértelmezés az 1, de ezt is szigoríthatjuk)*. A tanúsítványok automatikus kiadása *(AutoEnrollment)* csak akkor működik, ha ez az érték nem nagyobb egynél *(ezt az egy digitális aláírást a Windows automatikus tanúsítványkérés esetén még képes elhelyezni a kérésen – persze csak akkor, ha a tulajdonos rendelkezik a feltételeknek megfelelő aláíró tanúsítvánnyal)*.
A kérést aláíró tanúsítvánnyal szemben Application és/vagy Issuance Policy bővítmény-elvárásokat köthetünk ki. Az ábrán látható példa olyan aláíró tanúsítványt vár el, amit magas szintű biztonsági eljárás keretében adtak ki *(azaz az Issuance Level mezőjében a High Assurance OID-je szerepel)*, valamint a Certificate Request Agent nevű Application Policy-vel rendelkezik. *(Hozzátesszük, hogy az Issuance Policy elvárással rendelkező tanúsítványkérések esetén kifejezetten ezt az Application Policy-t kell használnunk – a neve is ez: tanúsítványkérés-aláíró ügynök –, különben magával a digitális aláírás végrehajtásával lesznek gondjaink)*.
Természetesen azt is megtehetjük, hogy Issuance Policy követelményeket nem, csak Application Policy igényeket támasztunk – ekkor a Certificate Request Agent helyett választhatunk mást is.

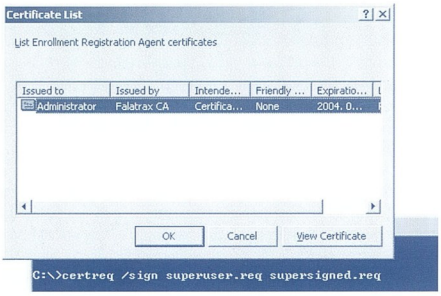
A dialógusablak alján még azt kell meghatároznunk, hogy a tanúsítvány megújítása esetén is leszünk-e pontosan ennyire szigorúak *(azaz a megújító kérést is aláíratjuk-e) (Same criteria as for enrollment)*, vagy megelégszünk azzal, ha a tulaj rendelkezik a tanúsítvány *(még éppen)* érvényes változatával *(Valid existing certificate)*.

A tanúsítványkérés digitális aláírása

Amikor tehát egy tanúsítványsablon digitálisan aláírt, „jóváhagyott" tanúsítványkérést definiál, a Windows webes felülete automatikusan fájlba menti a tanúsítványkérést. Ennek eredménye általában egy *.req fájl. Miután a tanúsítványkérés elkészült, ezt a fájlt el kell juttatnunk az aláíró személyhez *(valakihez, aki birtokolja az aláíráshoz szükséges paraméterekkel rendelkező tanúsítványt)*. Az aláírónak az alábbi parancsot kell kiadnia:

```
certreq /sign <eredeti.req> <alairt.req>
```

Az <eredeti.req> helyére értelemszerűen az eredeti kérésfájl, az <alairt.req> helyére pedig az aláírt eredményfájl nevét kell megadnunk. Ha az utóbbi paramétert nem adjuk meg, a megjelenő fájldialógusban menet közben is megtehetjük majd. Az aláíráshoz használt tanúsítványt a parancs futtatása során kell kiválasztanunk.



Tanúsítványkérés digitális aláírása

A Certificate List ablakban az elvárásoknak megfelelő tanúsítványaink jelennek meg *(ha a lista üres, nem rendelkezünk megfelelő tanúsítvánnyal)*. A tanúsítvány kiválasztása után az aláírt kérésfájl elkészül, amit immár elküldhetünk a Windows CA-nak, ha a webes felületen a fájlba mentett tanúsítványkérés küldését választjuk, és a megfelelő mezőbe bemásoljuk a fájl tartalmát. Ne felejtsük el kiválasztani újra a használni kívánt tanúsítványsablont!

Az digitálisan aláírt tanúsítványkérés elküldése *(ha csak egy aláírásra van szükség, és rendelkezünk a kérés aláírásához szükséges tanúsítvánnyal is)* a tanúsítványkezelő MMC konzolból egy kicsit egyszerűbb:

Az MMC konzolból egyszerűbb az aláírt tanúsítványkérés elküldése

A konzol Request New Certificate... parancsával indítható varázsló ugyanis automatikusan ellátja a kérést a megfelelő digitális aláírással.

A következő részben a tanúsítványkiadók hierarchiájával kapcsolatos újdonságokról lesz szó.

Folytatjuk...

Fülöp Miklós
mick@inetcom.hu

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

NAT-T: IPSec NAT-on keresztül

IPSec NAT Traversal – az IPSec új üzemmódja

A Windows 2000-ben bevezetett IPSec protokoll egyik nagy hátránya volt, hogy az általa titkosított csatorna nem volt képes áthaladni hálózati címfordító *(NAT – Network Address Translation)* eszközökön. A Windows 2003 már tartalmazza, Windows 2000-hez és XP-hez pedig letölthető azt a bővítés, ami megoldja ezt a problémát is.

A hálózati címfordításról címszavakban

Mi is az a NAT? A Network Address Translation, azaz „hálózati címfordítás" egy olyan eljárást takar, amelynek során egy alhálózat kijáratánál található átjáró a rajta áthaladó hálózati csomagok fejlécét és tartalmát úgy módosítja, hogy elrejti a csomag eredeti „származási" helyét, azaz a belső, védett hálózaton található számítógépek IP-címét. A belső IP-cím helyére az átjáró külső „lábának" IP címe kerül, a válaszként érkező csomagok pedig ugyanide érkeznek. A külső hálózatról beérkező csomagok eljuttatása a megfelelő belső címzetthez a NAT átjáró feladata *(többek között)*.

A hálózati címfordítás vázlata

Nézzük például a fenti ábrát: itt a belső hálózaton található Ügyfél elindítja a webböngészőjét, és csatlakozni próbál a külső hálózaton található webkiszolgálóhoz. Az Ügyfél belső IP címe egy privát cím, 192.168.0.11. A kapcsolathoz a böngésző az ügyfél operációs rendszerétől a 3873-as TCP port címet kapja. Az ügyfél számítógépét elhagyó csomag feladója tehát a következő „feladó" adatokat tartalmazza: 192.168.0.11:3873. A csomag eljut az alapértelmezett átjáróhoz, ami NAT feladatokat is ellát. A NAT pedig, mielőtt a csomagot továbbítaná, a benne található belső IP-címet a saját külső, nyilvános IP címére cseréli *(11.22.33.44)*. Miután az ügyfél által használt portcím *(3873)* az átjáróban egyáltalán nem biztos, hogy szabad, a router általában egy másik portcímet kér az operációs rendszertől *(példánkban az 5319-et kapja)*. A nyilvános hálózatra kilépő csomag feladója tehát: 11.22.33.44:5319.

Ezt az összerendelést az átjáró egy belső táblázatban szépen meg is jegyzi, hiszen a beérkező csomagok továbbításához erre szükség is lesz:

Belső IP-cím	Belső Port	Külső IP-cím	Külső Port
192.168.0.11	3873	11.22.33.44	5319

Példa egy NAT táblázatból

A csomag eljut a webkiszolgálóhoz *(22.33.44.55:80)*, aki a választ természetesen a feladónak küldi vissza: 11.22.33.44:5319. Az átjáró a beérkező csomag cél-IP és portcímét megkeresi a NAT-táblázatban, majd a megfelelő módosítások *(fejlécmódosítás)* után továbbítja a csomagot az eredeti feladónak *(192.168.0.11:3873)*.

A NAT táblázatbeli összerendelés TCP csatorna esetén egészen addig érvényes, amíg az ügyfelek le nem bontják a TCP csatornát. UDP kommunikáció esetén azonban nincs csatornaépítés, csak ad hoc közlekedő csomagok. Összerendelő táblázatbejegyzés természetesen UDP csomagok esetén is készül a NAT-ban, de ezek a bejegyzések, adott idő *(Time-Out)* múlva automatikusan megszűnnek.

Ez azonban csak a jéghegy csúcsa: a NAT eszköz a fejlécek cseréberélésén és a csomagok továbbításán kívül még sok minden mást is művel. Vannak többek között olyan protokollok, amelyek nemcsak a csomagok fejlécében, de magukban az átvitt adatokban *(a kommunikációban)* is továbbítják az IP-cím információkat *(ilyen például az FTP is)*. Ha a NAT csak az FTP csomagok fejlécét cserélgetné, a csomagokon belül található címek a belső címek maradnának, és az FTP nem működhetne. Ezért a NAT felismeri és módosítja az ilyen kényes protokollok tartalmát is *(ez már egyébként az Application Layer Gateway, ALG feladata)*. Ennyi *(rövid és felületes)* bevezető után már itt az ideje rátérnünk az IPSec és a NAT problémájára.

Mi a baj az IPSec-kel?

Az előző bekezdés elolvasása után már nagyjából lehet sejteni, mi a probléma az IPSec csomagokkal. A probléma sokrétű *(akit részletesen érdekel, az **[ipsecnatreq]** címről letölthető dokumentumban elolvashatja)*. Néhány példa:

1. Az ISAKMP/Oakley protokoll

Az IPSec csatorna kiépítését az ISAKMP/Oakley protokoll segítségével végzik a kommunikációban majdan résztvevő számítógépek. Az ISAKMP protokoll tartalmazza a csatorna két végpontjának IP-címét *(ami NAT esetén természetesen megváltozik)*.

2. Az ESP protokoll

A titkosított csatorna kiépítése után a titkosított IPSec kommunikáció ESP csomagok formájában halad a hálózaton. Ezek a csomagok csak az IP-címeket, illetve egy belső azonosítót *(SPI)* tar-

talmaznak. Amikor a csomag eljut a címzetthez, az az SPI alapján dönti el, hogy a csomag melyik alkalmazáshoz tartozik *(az SPI nagyjából a TCP/UDP portcím szerepét tölti be)*. Emellett az SPI utal természetesen magára az IPSec „csatornára" is. Egy számítógép ugyanis egynél több IPSec csatornát is kiépíthet, ugyanazzal a számítógéppel, vagy más partnerekkel, és minden IPSec csatorna beállításai *(azonosítási mód, titkosítás, a kulcsmenedzsment értékei)* csatornánként különbözőek is lehetnek.

Pillantsunk vissza a NAT vázlatára! Ha a belső hálózatból egyidejűleg több ügyfél próbálna IPSec-kapcsolatot építeni a kiszolgálóval *(még ha az ISAKMP miatt sikerülne is)*, a NAT nem lenne képes feldolgozni a beérkező ESP csomagokat *(nem tudná, melyik belső ügyfélnek továbbítsa azokat)*. Az SPI-érték a NAT számára nem mond sokat, ugyanis az a különböző „irányba" haladó csomagok esetén különböző.

3. Az AH protokoll

Ha az IPSec kommunikáció nem titkosított, csak a csomagok integritásvédelmét *(digitális aláírását)* vesszük igénybe, az IPSec hálózati forgalom ESP helyett AH *(Authenticated Header)* csomagok formájában halad a hálózaton. Az AH csomag digitális aláírása azonban tartalmazza a feladó és címzett IP-címét, a csomag fejlécének megváltoztatása tehát elrontja a csomagok digitális aláírását, és érvényteleníti azokat.

A megoldás

A megoldás kétrétű: egyrészt, az ISAKMP protokollt kell felkészíteni a NAT kezelésére *(az ISAKMP protokoll az UDP 500-as portot használja)*, másrészt pedig az ESP csomagok továbbítását és pontos célbajuttatását kell megoldanunk. *(A specifikáció jelenleg az AH csomagok átvitelét nem támogatja.)* Az első feladat viszonylag egyszerű, az UDP protokoll ugyanis önmagában elviseli a NAT-ot *(kivéve azt az apróságot, hogy az UDP-összerendelések a NAT-táblából egy idő után szeretnek törlődni)*. Az ISAKMP esetében tehát gyakorlatilag csak a protokoll bővítésével kell foglalkozni.

Az ESP csomagok védelme kicsit összetettebb, de a megoldás gordiuszi: ha az ESP nem viseli el a NAT-ot, az UDP viszont igen, miért ne csomagolhatnánk be az ESP csomagokat UDP-ruhába a továbbítás idejére? Ez lesz az UDP 4500-as port, amibe – mint majd látni fogjuk – az ISAKMP protokoll jó része is beköltözik majd.

A megoldáscsomagot NAT-Traversalnak *(NAT-T, NAT-átjárás)* nevezték el.

Az ISAKMP bővítményei

Az ISAKMP protokoll az UDP 500-as portcímet használja, ráadásul az eredeti IPSec-implementációban úgy, hogy a csomagok feladó és portcíme egyaránt az UDP 500 volt. Ha belegondolunk egy kicsit, ez már önmagában gondot okoz a NAT-nál, ugyanis a NAT mögül kiküldött ISAKMP csomag portcíme legkésőbb a második számítógép próbálkozásakor már foglalt lesz.

Az első lépés tehát az volt, hogy mostantól a hálózati eszközöknek és számítógépeknek akkor is fogadni és kezelni kell az ISAKMP csomagokat, ha azoknak a feladó port címe nem, csak a címzett port címe 500. Ha például az ISAKMP csomag a feladó UDP 12345-ös portcíméről a címzett UDP 500-as portjára érkezik, azt fel kell dolgozni, a válaszokat pedig *(értelemszerűen)* a feladó UDP 12345-ös portjára kell küldeni.

A következő lépés annak kiderítése, hogy az IPSec-csatornát építő felek közül ki ismeri a NAT-T üzemmódot. Ezt a tagok nagyon egyszerűen, az első ISAKMP csomagba kölcsönösen elhelyezett, előre definiált Vendor-ID bájtsorozat elhelyezésével illetve kiolvasásával döntik el.

A NAT felismerése: NAT-D

A NAT-T kompatíbilis ISAKMP protokoll képes felismerni, ha az adatátviteli csatorna címfordításon halad keresztül. Az ISAKMP harmadik *(válaszként pedig negyedik)* csomagjában mindkét fél elküldi a saját illetve másik tag *(általa ismert)* IP-címéből és portcíméből készült hash-értéket. *(Ha a gépnek több saját IP-címe van, mindegyikhez készül egy-egy NAT-D hash)*. Ha a Windows 2003 Network Monitorával megnézünk egy ilyen forgalmat, láthatjuk is az ISAKMP NAT-Discovery részét:

```
⊞ FRAME: Base frame properties
⊞ ETHERNET:  EType = Internet IP (IPv4)
⊞ IP: Protocol = UDP - User Datagram; Packet ID = 53602;
⊞ UDP: Src Port: isakmp (500); Dst Port: isakmp (500); L
⊟ ISAKMP: Major Version: 1  Minor Version: 0  Length: 23
    ISAKMP: Initiator cookie = 7B 74 B9 96 41 F6 CF 08
    ISAKMP: Responder cookie = 19 EA BE 45 0A 9E AF 20
    ISAKMP: Next payload = Key Exchange
    ISAKMP: Major version = 1 (0x1)
    ISAKMP: Minor version = 0 (0x0)
    ISAKMP: Exchange type = Identity Protection
  ⊞ ISAKMP: Flags summary = 0 (0x0)
    ISAKMP: Message ID = 0 (0x0)
    ISAKMP: Length = 232 (0xE8)
  ⊞ ISAKMP: Payload type = Key Exchange
  ⊞ ISAKMP: Payload type = Nonce
  ⊟ ISAKMP: Payload type = NAT Discovery
      ISAKMP: Next payload = NAT Discovery
      ISAKMP: Reserved = 0 (0x0)
      ISAKMP: Payload length = 24 (0x18)
      ISAKMP: NAT Discovery data = D0 C9 E7 53 36 FC E7
  ⊞ ISAKMP: Payload type = NAT Discovery
```

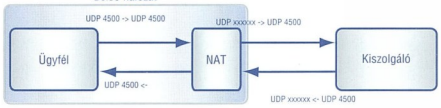
NAT-D hash az ISAKMP csomagban

Miután a csomag megérkezik, a címzett elvégzi ugyanezeket a hashműveleteket az általa ismert címekkel. Ha a két hash nem egyezik, az IP-cím vagy a portcím a csomag továbbítása közben megváltozott, tehát a kommunikáció során valahol címfordítás történt *(sőt, még az is kiderül, hogy melyik félnél)*.

A NAT-T kommunikációs port *(ISAKMP2, UDP 4500)*

Miután kiderült, hogy a NAT-T-re szükség van, a korábban említett okok miatt a rendszer áttér az UDP 4500-as port használatára *(ettől a pillanattól kezdve a teljes kommunikáció során az ISAKMP már itt utazik, az UDP 500-as port helyett)*. Ezt egy *(Windows 2003 Server-beli)* Network Monitor segítségével szépen lehet is követni.

Az ISAKMP2 csomagok a feladó UDP 4500-as portjáról a célgép UDP 4500-as portjára indulnak el, de a feladó végleges portcíme *(és persze IP címe)* a NAT miatt ettől végül eltérő lesz:



Az ISAKMP2 által használt portcímek

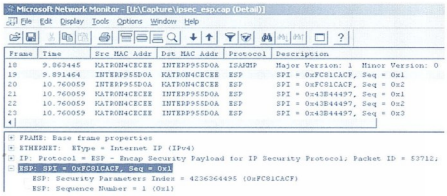
A kiszolgáló válasza értelemszerűen a NAT által módosított portcímre érkezik, ami aztán a NAT-táblázat alapján jut el a belső ügyfélhez.

A NAT Keep-Alive csomag

A felépített csatorna védelme érdekében a NAT-T üzemmódban futó ISAKMP adott időnként *(20 másodpercenként)* úgynevezett NAT Keep-Alive csomagot küld. Ez egy olyan ISAKMP2 UDP-be ágyazott „félkész" ESP-csomag, amely egyetlen bájtot tartalmaz *(értéke $FF)*. Ezt a csomagot az IPSec figyelmen kívül hagyja, a NAT átjáró viszont ennek köszönhetően életben tartja az amúgy illékony UDP-összerendelési bejegyzést a NAT-táblában.

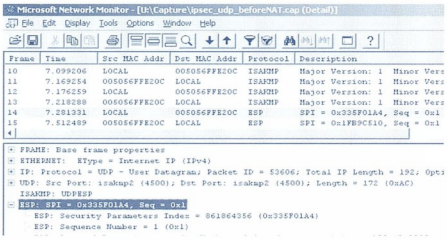
Az UDP-be ágyazott ESP forgalom

Végül elérkeztünk a lényeghez: a sok körítés mellett-után megkezdődhet a titkosított IPSec kommunikáció! Az ESP csomagok ISAKMP2 csomagba ágyazva utaznak a hálózaton. Lássunk egy Network Monitor részletet egy NAT-T nélküli ESP csomagról:



Az eredeti megoldás: IPSec NAT-T nélkül

Figyeljük meg, hogy az ESP réteg közvetlenül az IP protokoll után következik. Vessük össze most ezt egy NAT-T üzemmódban zajló forgalommal:

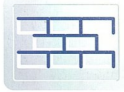


Az ESP új ruhája: NAT-T üzemmódban UDP-be csomagolva

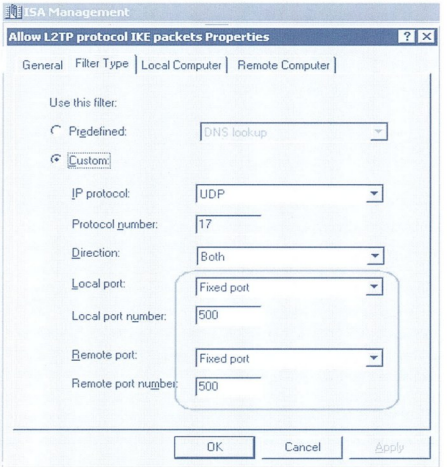
Ha jól megnézzük az ábrát, észrevethetjük, hogy az IP és az ESP közé egy UDP réteg ékelődött be *(talán nem meglepetés, hogy az UDP 4500-as portcímen)*. Ennek a csomagolásnak *(no és persze az előző két oldalon leírtaknak)* köszönhetően az ESP vígan közlekedik keresztül a címfordításon is.
(Ha a fenti csomagokat nem Windows 2003-beli Network Monitorral nézzük, a NetMon az első négy (UDP 500-as porton haladó) ISAKMP csomag kivételével csupa ismeretlen UDP forgalmat jelenítene meg.)

Tűzfalkonfiguráció

Az új kommunikációs port bevezetése azt jelenti, hogy a NAT-T IPSec támogatásához módosítanunk kell az esetleg jelen lévő tűzfalaink beállításait. Az előző oldalon leírtuk az ISAKMP portcímeinek alakulását *(feladó: UDP xxxxx, címzett: UDP 4500)*. Ehhez új csomagszűrőt kell engedélyeznünk. Emellett ellenőrizzük, hogy az eredeti ISAKMP forgalom *(ami UDP 500-as feladó és címzett portcímmel zajlott)* szűrője engedélyezi-e az UDP 500-tól eltérő portcímről beérkező csomagokat is. Az ESP csomagok átvételének támogatására viszont már nem lesz szükség.



Az ISA Server például egy parancs segítségével automatikusan képes létrehozni az eredeti *(L2TP over)* IPSec implementációhoz szükséges csomagszűrőket. Ezek közül az „Allow L2TP Protocol IKE Packets" csomagszűrő beállításait kell módosítanunk, attól függően, hogy a tűzfal mögötti kiszolgáló fogadja vagy kezdeményezi a kapcsolat felépítését. Ha mindkettőt, akkor újabb csomagszűrő szabályt is kell létrehoznunk.



Az ISA Server automatikusan létrehozott csomagszűrőjét módosítanunk kell

Emellett természetesen létre kell hoznunk azokat az új csomagszűrőket is, amelyek az új ISAKMP2 protokollt engedik át. Összefoglalva tehát:

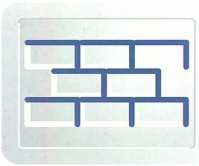
IPSec NAT-T kapcsolat	Hely portcím	Távoli portcím
Bejövő ISAKMP	UDP 500	UDP összes
Bejövő ISAKMP2	UDP 4500	UDP összes
Kimenő ISAKMP	UDP összes	UDP 500
Kimenő ISAKMP2	UDP összes	UDP 4500

Támogatás

Az IPSec NAT-Traversal a Windows Server 2003-ban már megtalálható. Windows 2000-hez és Windows XP-hez a **[w2kxp]** címről, a 818043-as Tudásbázis-cikkből tölthető le. A cikk írásának pillanatában a Windows XP-re írt frissítés átmenetileg nem érhető el, de reménykedjünk, hogy ami késik, az nem múlik. Megéri várni!

Fülöp Miklós
mick@inetcom.hu

További információkért érdemes meglátogatni az IETF IP Security Working Group weboldalát az alábbi webcímen:
http://www.ietf.org/html.charters/ipsec-charter.html



Biztonságos aláíráskezelő alkalmazás (BALE) készítése IV.

Adatok és komponensek

A hosszúra nyúlt bevezetés után végre eljutottunk a gyakorló informatikus kollégák számára minden bizonnyal kedvesebb témákhoz. Jelen részben az aláíráskezelő alkalmazás által használt adatokról, az aláíráslétrehozó alkalmazás ezeket hasznosító komponenseiről, valamint az alkalmazással szemben támasztott követelményekről lesz szó. Egyben folytatódik a mindennek nevet adunk-szakkör, ami most ér igazi csúcspontjára, hiszen most egyes fogalmaknak új elnevezést adunk. Tessék kapaszkodni, mert aki eddig netán értette, az most könnyen elveszítheti a fonalat.

Mielőtt belekezdenék azért megjegyzem, hogy ezt az átkeresztelési játékot nem személyes antiszociális indíttatásból teszem, hanem mert ez az „iparági gyakorlat". Az egyes külön dolgozó szabványügyi szervezetek, csoportok, cégek mind saját fogalmakat alkotnak ugyanazon dolgokra, néha házon belül is többet, attól függően, hogy a téma mely részére koncentrálnak épp. Jobbnak láttam, ha nem egységesítem ezeket az elnevezéseket, inkább használom az eredetieket, jelölve, hogy máshol, máskor miként nevezik ugyanazt. Így felvérteződve már nem érheti nagy meglepetés azokat, akik más forrásból is tájékozódnak.

Az aláíráskezelő alkalmazás adatai

Az aláíráskezelő alkalmazás egyes komponensei a következő adatokat és információs egységeket kezelik:

- Aláírói dokumentum: az a bármilyen tartalmú dokumentum, melyet az aláíró el kíván látni az aláírásával.
- Aláírásjellemzők: az aláírás olyan kiegészítő adatai, amelyek az aláírási szabályzatnak megfelelően az aláírói dokumentummal együtt aláírásra kerülnek *(lásd tételesen később)*.
- Aláírandó adat alkotóelem: az aláírásjellemzők és az aláírói dokumentum önállóan.
- Aláírandó adat: az aláírói dokumentum, az aláírásjellemzők és opcionálisan egyéb aláírásra kerülő információk *(ld. később)* együtt.
- Formattált aláírandó adat: az aláírandó adat az aláírási szabályzat és/vagy az alkalmazás előírásai szerint formattálva, az aláírandó adat formattáló által *(pl. XML Advanced Electronic Signature formátumra)*.
- Aláírandó adat képviselő: a formattált aláírandó adat lenyomata, melyet a lenyomatoló komponens állít elő, majd formattált *(feltöltött és szabványos formára hozott)*. Ezt az adatot küldi az alkalmazás a BALE-nak aláírásra.
- Aktiválási adat: a BALE aktiválására, illetve az aláíró BALE felé történő hitelesítése szolgáló adat.
- Minősített elektronikus aláírás: Az aláíró algoritmussal a BALE által kódolt aláírandó adat képviselő *(korábban digitális aláírásnak volt nevezve)*.
- Aláírt adathalmaz: a minősített elektronikus aláírás és a formattált aláírandó adat mezői, egyéb aláírással nem ellátott információkkal együtt, melyeket az aláírt adathalmaz összeállító állított össze az aláírt adathalmaz típusnak megfelelően. Ide kerülhet az aláírás értékén végzett időbélyegző, mely az aláírás idejét hivatott igazolni. Ez az aláíráslétrehozó alkalmazás kimenete, amit korábban elektronikus aláírásnak hívtunk.
- Aláírt adathalmaz-típus: az aláírt adathalmaz típusa, amely meghatározza az aláírt adathalmaz tartalmát és formátumát.
- BALE-tulajdonos azonosító: a BALE-tulajdonosának azonosító adata *(jellemzően neve)*, mely a BALE felületén *(pl. kártyán dombornyomottan)* vagy a belsejében elektronikusan van feltüntetve/tárolva.
- BALE-információ: BALE-tulajdonos azonosító és egyéb BALE adatok.

Az aláírás megtétele során felhasználásra kerül az ún. aláíráslétrehozó adat is, mely *(immár újabb definícióval)* a BALE által az aláírandó adat képviselő kódolására felhasznált adat. Ezt az adatot az alkalmazás *(közvetlenül)* nem kezeli.

Az aláírásjellemzők a következők lehetnek:

- Aláíró tanúsítványa: az aláírás megtétele során az aláírásban résztvevő aláíráslétrehozó adat kiválasztására szolgál *(különösen amennyiben aláíró több aláíráslétrehozó adattal rendelkezik)*. Az aláírás ellenőrzésekor az aláírás értékének dekódolására, valamint a nyilvános kulcs aláíróhoz tartozásának és érvényességének megállapítására használatos.
- Aláírási szabályzat referencia: utalás az aláírási szabályzatra, mely alapján az aláírás készült, s mely alapján az ellenőrizhető és értelmezhető.
- Kötelezettségvállalás típusa: az aláíró által az aláírással kifejezni kívánt szándék az aláírási szabályzatnak megfelelően *(amennyiben az aláírási szabályzat több ilyet megkülönböztet)*.
- Tartalomformátum: az aláírói dokumentum formátuma, mely az aláíró és ellenőrző általi megtekintést és használatot meghatározza.
- További aláírásjellemző lehet bármi, amit az aláírási szabályzat vagy más előírás meghatároz, pl. időbélyegző, archív tanúsítványok.

Egyéb aláírásra kerülő információk a következők lehetnek:

- Az aláírás ideje az aláíró állítása szerint.
- Az aláíró kinyilvánított szerepköre.
- Az aláíró hitelesített szerepköre *(pl. attribútum tanúsítvány).*
- Az aláírás fizikai helye az aláíró állítása szerint.
- Időbélyegző az aláírói dokumentumról vagy más adatról *(ami nem az aláírás idejét tanúsítja, bár arra bizonyíték, hogy az aláírás csak az időbélyegzőben szereplő időpont után jöhetett létre).*
- Bármilyen további adat, amit az aláírási szabályzat meghatároz.

Az aláíráslétrehozó alkalmazás komponensei

A bemutatott adatokat az aláíráslétrehozó alkalmazás következő komponensei kezelik. A komponensek két kategóriába, a megbízható összetevők és az alkalmazásspecifikus összetevők csoportjába oszthatók. A megbízható összetevőknek minden aláíráslétrehozó alkalmazásban megtalálhatónak kell lenni. Az alkalmazásspecifikus összetevők jelenléte feltételes, felépítésük és működésük az alkalmazási környezettől és körülményektől függ.

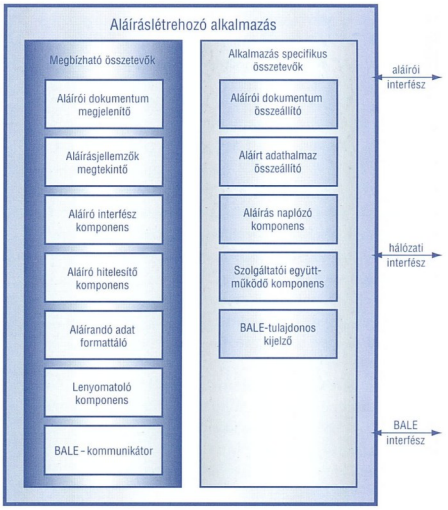
Az aláíráslétrehozó alkalmazás megbízható összetevői a következők:

Aláírói dokumentum megjelenítő
Aláírásjellemzők megtekintő
Aláíró interfész komponens
Aláíró hitelesítő komponens
Aláírandó adat formattáló
Lenyomatoló komponens
BALE-kommunikátor

Az alkalmazásspecifikus összetevők az alábbiak:

Aláírói dokumentum összeállító
Aláírt adathalmaz összeállító
Aláírásnaplózó komponens
Szolgáltatói együttműködő komponens
BALE-tulajdonos kijelző

Aki tervezett vagy fejlesztett már aláíráskezelő alkalmazást, az bizonyára találkozott az említett adatok és komponensek némelyikével. Csak nem nevezte és különítette el őket így. Tervezés és *(tanúsítást megelőző)* dokumentálás előtt mindenképp érdemes hasonló módon komponensekre bontani az alkalmazást és a kezelt adatokat. Azokat a műveletek, amelyek az egyes komponensek funkcionalitását kielégítik, szinte biztosan ott lesznek az alkalmazásban, így nem jelent járulékos fejlesztési feladatot e komponensek megvalósítása. Jobb esetben csak egy új szemléletmódot kell magunkévá tennünk, mely biztosan kifizetődő.
Az egyes funkciók logikai és fizikai elkülönítése, ha nem is növeli közvetlenül az alkalmazás biztonságát, jobban áttekinthető, könnyebben tesztelhető rendszert eredményez, melynek tanúsításával is kevesebb gondunk lesz.



Aláíráslétrehozó alkalmazás komponensei

A működés áttekintése

A következő ábrán az aláírás létrehozásának menetét szemléltetem, az egyes komponensek szerepének, az egyes adatok, egyéb külső szereplők, a BALE és az aláíró közreműködésének feltüntetésével *(a komponensek téglalapban, az alkalmazásspecifikus komponensek szaggatott vonallal, a kezelt adatok rombuszban, az egyéb külső szereplők hatszögben, a BALE és az aláíró közreműködése ellipszisben, a lépések sorszámozásával van jelölve).*

Aláírói dokumentum megjelenítő

Az aláírói dokumentum megjelenítő feladata, hogy az aláírói dokumentum összeállító által előállított aláírói dokumentumot a tartalomformátumnak megfelelő módon megjelenítse az aláíró számára. Ennek célja, hogy az aláíró megbizonyosodhasson arról, hogy valóban azt írja alá, amit szándékozik *(amit az aláíró interfész komponenssel kiválasztott).* Az összeállító és megjelenítő funkciók szétválásának az oka, hogy az összeállító általában nem képes a megjelenítő biztonsági kritériumainak teljesítésére. Az aláírói dokumentum megjelenítő viszont minden bizonnyal csak véges számú tartalomformátumot kezel.
Az aláírói dokumentum összeállító és megjelenítő elvileg megegyezhet: vagy azon a módon, hogy az aláírói dokumentum összeállító egyben aláíráslétrehozó alkalmazás is *(amihez az aláíráslétrehozó alkalmazás összes biztonsági kritériumát ki kell elégítenie),* vagy úgy, hogy az aláíráslétrehozó alkalmazás az aláírói dokumentum összeállítót használja aláírói dokumentum megjelenítőként *(mely esetben csak a megjelenítő biztonsági kritériumait kell kielégítenie).*

Aláírásjellemzők megtekintő

Az aláírásjellemzők megtekintő feladata, hogy az aláíró számára megjelenítse az általa *(az aláíró interfész komponenssel)* kiválasztott és/vagy az aláíráslétrehozó alkalmazás által az

aláírandó adathoz rendelt összes aláírásjellemzőt, abból a célból, hogy aláíró számára egyértelművé tegye az aláírói dokumentum értelmezését és az aláírás által kifejezett szándékot. Ez a komponens jeleníti meg például a kiválasztott kötelezettségvállalás típusát és az aláíró tanúsítványát.

Aláíró interfész komponens

Az aláíró interfész komponens feladata, hogy illesztőfelületet képezzen az aláíró és az aláíráslétrehozó alkalmazás között, abból a célból, hogy az aláíró számára irányíthatóvá és ellenőrizhetővé tegye az aláírás előállítási eljárást *(pl. aláírás jóváhagyása)*, valamint, hogy az aláíráslétrehozó alkalmazás számára lehetővé tegye az aláíró felé történő jelzéseket *(hibajelzés, figyelmeztetés, kérdések stb.)*.
Az aláíró interfész komponens illetékességi körébe tartozik minden alkalmazás-aláíró kommunikáció, kivéve az aláíró BALE felé történő hitelesítését.

Aláíró-hitelesítő komponens

Az aláíró-hitelesítő komponens feladata, hogy az aláíró BALE felé történő hitelesítéséről gondoskodjon. Ennek egyik módja, hogy az aláírótól elkért BALE aktiválási adatot összehasonlítja a BALE-n tárolttal. Megvalósulhat a PIN kód billentyűzeten keresztüli bevitelével, biometrikus azonosítással és egyéb módszerekkel is. Az alkalmazott módszertől függően szükség szerint igénybe veszi a BALE-kommunikátort.

Aláírandó adat formattáló

Az aláírandó adat formattáló feladata, hogy az aláírói dokumentumból és az aláírásjellemzőkből előállítsa a formattált aláírandó adatot, a kiválasztott aláírási szabályzatnak és/vagy az aláíráslétrehozó alkalmazás képességeinek megfelelően.

Lenyomatoló komponens

A lenyomatoló komponens feladata az aláírandó adat képviselő előállítása a formattált aláírandó adatból, és ennek formattálása. Ez a gyakorlatban a lenyomat generálását, szükség szerint feltöltését *(padding)* és előírások szerinti formattálását jelenti *(pl. PKCS #1, ISO-IEC 9796-2 formátumba)*. A lenyomat generálásában és feltöltésében képességei szerint részt vehet az aláíráslétrehozó eszköz is.

BALE-kommunikátor

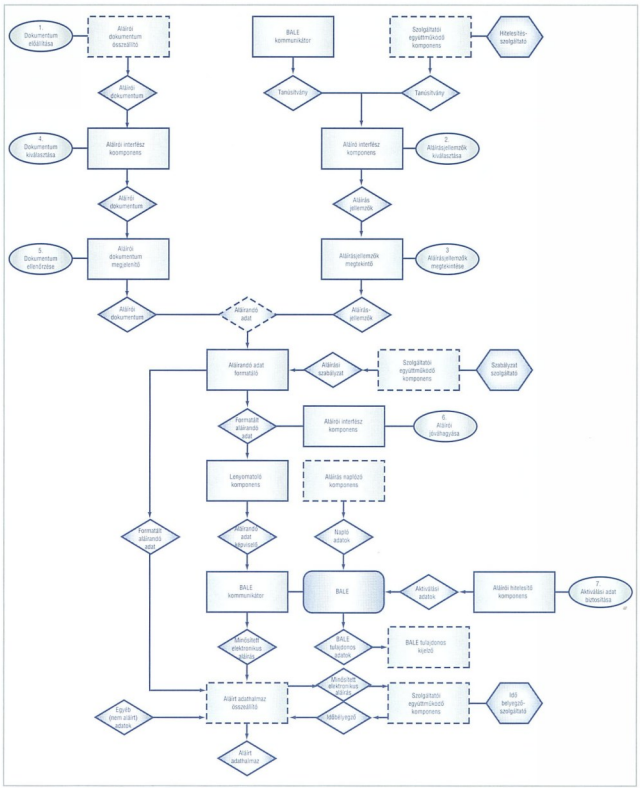
A BALE-kommunikátor feladata, hogy az aláíráslétrehozó alkalmazás és a BALE közti adatcserét kezelje, ennek biztonságáról gondoskodjon.

Aláírói dokumentum összeállító

Az aláírói dokumentum összeállító általában egy szövegszerkesztő *(pl. Word)*, de lehet más dokumentumkészítő program is *(táblázatkezelő, prezentációkészítő stb.)*, sőt az aláírási szabályzatban meghatározott tartalomformátumokat előállító bármilyen alkalmazás is. Feladata, hogy előállítsa az aláírói dokumentumot.

Aláírt adathalmaz összeállító

A komponens feladata, hogy az aláírt adathalmazt összeállítsa az aláíró által kiválasztott aláírt adathalmaz típusnak megfelelően.



Aláírás létrehozása

Inputja a minősített elektronikus aláírás és opcionálisan a formattált aláírandó adat, illetve annak mezői. Ezen kívül egyéb, aláírással el nem látott információkat is tartalmazhat. Az alkalmazásnak érdemes rejtjelezési képességgel is rendelkeznie az aláírt adathalmaz bizalmasságának biztosítására.

Aláírásnaplózó komponens

Az Aláírásnaplózó komponens az aláíráslétrehozó eszközön naplózza az egyes aláírások legfontosabb adatait *(aláírás, lenyomat, időpont, aláírói dokumentum azonosító, alá-*

írásjellemzők, stb.). Ennek célja, hogy az aláíró a napló adataiból észrevegye az eszköz használatával kapcsolatos esetleges visszaéléseket *(hogy valaki más használta aláírásra)*, valamint képes legyen saját aláírásainak későbbi azonosítására.
Az aláíráslétrehozó eszközök erősen limitált memóriája természetesen csak véges számú naplóbejegyzés támogatására képes, ami a gyakorlatban ciklikus naplózást jelent, a legrégibb naplóbejegyzések mindenkori felülírásával.
Az aláíráslétrehozó alkalmazás szerepe, hogy az aláíráslétrehozó eszköz naplózási funkcióját ellássa az ehhez szükséges adatokkal *(pl. aláírói dokumentumazonosító, aláírásjellemzők)*, illetve az eszközön elvégezze a naplózást, amennyiben az eszköz ilyen beépített funkcióval nem rendelkezik. Amennyiben az eszköz így sem teszi lehetővé a naplózást, ajánlott legalább az elvégzett aláírások számának naplózása aláíráslétrehozó adatonként, illetve tanúsítványonként.
Az aláíráslétrehozó alkalmazásnak biztosítania kell a naplózott adatok megtekintésének lehetőségét is az aláíró interfész komponensen keresztül.

Szolgáltatói együttműködő komponens

A komponens az elektronikus aláírási szolgáltatókkal *(hitelesítésszolgáltató, időbélyegző szolgáltató)* való együttműködéséről gondoskodik. Feladata többek között tanúsítványok importálása *(szolgáltatótól)*, érvényességük ellenőrzése, időbélyegző kérése. A tanúsítványok letöltése és érvényességének ellenőrzése az aláírói dokumentumban szereplő egyéb aláírásokkal kapcsolatban *(többszörös aláírások)*, illetve aláíró saját tanúsítványával kapcsolatban lehet szükséges *(amennyiben a BALE azt nem tárolja)*.

BALE-tulajdonos kijelző

A BALE-tulajdonos kijelző komponensnek a BALE-n elektronikusan tárolt tulajdonos azonosító kijelzéséről kell gondoskodni, az aláíró interfészkomponensen keresztül. Ez olyan esetekben különösen fontos, mikor az eszközön nem lehetséges a név hagyományos írott formában történő feltüntetése *(pl. USB token, HSM)*.

Általános követelmények

Az aláíráslétrehozó alkalmazás tervezésekor és fejlesztésekor számos biztonsági és funkcionális követelményt kell a szakembereknek figyelembe venni. Az összes követelmény megismertetéséről terjedelmi okok miatt le kell mondanom, de a fontosabbakat és érdekesebbeket bemutatom, a követelményt indokoló veszélyekkel, és néhol a lehetséges megoldásokkal együtt. A biztonsági követelmények között vannak általánosak, melyek az alkalmazás egészére vagy összes moduljára vonatkoznak, valamint az egyes adatokra és komponensekre specifikusak.

Az általános biztonsági követelmények közé sorolhatóak a következők:

- Garantálni kell az alkalmazásnak, a komponensek, konfigurációs adatok és paraméterek sértetlenségét, illetve az ezekben történő *(vétlen vagy rosszindulatú)* változások felismerhetőségét. A követelmény különösen vonatkozik az alkalmazás online letöltődő elemeire.
  A követelmény magáért beszél: amennyiben egy behatoló *(vagy pl. vírus)* képes az alkalmazás vagy annak valamilyen működést befolyásoló elemének módosítására, az hamis aláírás előállítását eredményezheti.
  A megoldás az lehet, hogy az alkalmazást magát, illetve annak komponenseit *(EXE, DLL, JAR, stb. állományok)* alá kell írni, oly módon, hogy azt a futtató környezet képes legyen ellenőrizni. Az online letöltődő elemek *(pl. plugin, BALE-meghajtóprogram, scriptek)* sértetlenségéről és hitelességéről szintén gondoskodni kell *(pl. SSL kapcsolat)*. Kerülni kell, ha mégis felhasználásra kerül, védeni kell az összes INI, CFG és egyéb külső állományokat, registry bejegyzéseket.
- Garantálni kell az alkalmazás által kezelt aláírásspecifikus adatok sértetlenségét.
  E nélkül egy behatoló az egyes adatokat az alkalmazás egyes komponenseiben, illetve az azok közötti kommunikációban módosíthatná.
- Garantálni kell az alkalmazás és a BALE közti kommunikáció sértetlenségét.
  Különben egy behatoló az alkalmazás és a BALE közötti kommunikációba belenyúlhatna, és azt módosíthatná.
- Garantálni kell az alkalmazás adatainak, különösen az aktivizáló adatnak a bizalmasságát *(az előző követelmény csak a sértetlenségről szólt)*.
  Amennyiben egy illetéktelen az adatok megismerésével bizalmas információk birtokába juthat, a megismert aktivizáló adatot később aláíráshamisítására használhatja fel, amennyiben az aláíráslétrehozó adathoz hozzáfér.
- Garantálni kell, hogy a felhasználó által kiválasztott adatok *(pl. aláírandó adat, aláírandó adat alkotóelemek)* jelenjenek meg a felhasználó előtt, majd ezek az adatok kerüljenek felhasználásra az aláíráslétrehozás folyamatában.
  Ha egy behatoló a kiválasztás és megjelenítés, illetve a felhasználás folyamatába képes beavatkozni, semmi nem garantálja, hogy az aláíró által látott, kiválasztott és jóváhagyott adatok kerülnek aláírásra, illetve az aláírás során felhasználásra.

Az aláírandó adat az alkalmazás által létrehozott és kezelt legkomplexebb objektum, amire vonatkozóan több követelménynek is eleget kell tenni. Így például a következőknek:

- Az aláírandó adatnak tartalmazni kell aláírói dokumentumot.
  Az aláírás nem jöhet létre egy üres dokumentumra, illetve null értékre.
- Az aláírandó adatnak tartalmazni kell az aláíró azon tanúsítványát *(vagy egy arra történő egyértelmű utalást)*, mely az aláírás készítésében részt vevő aláíráslétrehozó adathoz kapcsolódik *(különösen amennyiben aláíró több ilyennel rendelkezik)*.
  Ennek az a célja, hogy ne fordulhasson elő, hogy az ellenőrző aláíró egy másik tanúsítványát használja az aláírás ellenőrzéséhez.
- Az aláírandó adatnak tartalmazni kell az aláírói dokumentum formátumát, amennyiben az aláírási szabályzat több tartalomformátumot is meghatároz.
  Ne fordulhasson elő, hogy az ellenőrző az aláírói dokumentumot rosszul értelmezi, mert pl. nem megfelelő módon jeleníti azt meg.

Követelmények komponensenként

Az aláírói dokumentummegjelenítő feladata a nevében van. A komponenssel szemben támasztott követelmények jó része a tartalomformátummal összefüggésben értelmezett:

- Lehetővé kell tenni a tartalomformátum belefoglalását az aláírói dokumentumba, az aláírási szabályzatba vagy az aláírásjellemzőkbe.
- Az aláírói dokumentum megjelenítő által kezelt tartalomformátumokat ismertetni kell a felhasználókkal. Ez történhet a termék csomagolásán, dokumentációban, helpben.
- Az aláírót figyelmeztetni kell, amennyiben olyan tartalomformátumot választ, melyet az aláírói dokumentummegjelenítő nem kezel.

Mint látható, a követelmények nem kívánnak mindig bonyolult fejlesztési megoldásokat. Sokszor elégséges lehet egy egyszerű funkció, egy dokumentálás vagy csak egy figyelmeztető rendszerüzenet is. Az aláírói dokumentummegjelenítővel szemben további követelmények nem kötődnek a tartalomformátumhoz:

- Figyelmeztetni kell az aláírót, amennyiben az aláírói dokumentum egyéb aláírásokat is tartalmaz, és lehetővé kell tenni az aláíró számára ezen aláírások ellenőrzését.
  Ez a követelmény a többszörös aláírásokkal kapcsolatban értelmezett, amikor az aláíró egy olyan aláírói dokumentumot ír alá, amin már szerepel mások aláírása. Ezek között lehetnek nem valódi vagy nem érvényes aláírások is, mely esetben a dokumentum újabb aláírását ki kell zárni.
- Meg kell akadályozni, hogy aláíró az aláírói dokumentumot a megjelenítés során módosítsa.
  Az aláírói dokumentumot a megjelenítés előtt véglegesíteni kell. A komponens tényleg csak a megjelenítésre való. Amennyiben a megjelenítés során derül ki a dokumentum valamely hiányossága, az aláírás folyamatát meg kell szakítani. A követelmény egyaránt vonatkozik a véletlen és szándékos módosítások lehetőségére.

Az aláírásjellemzők megtekintőre jellemző elvárás az alábbi:

- Lehetővé kell tenni az aláíró számára az aláírásjellemzők közé foglalt tanúsítvány adatainak megtekintését.

Ennek az a célja, nehogy tévedésből nem megfelelő tanúsítvány kerüljön aláírásra, valamint a hozzá tartozó magánkulcs felhasználásra. Jellemző eset például, hogy az aláírónak több tanúsítványa is van, amik csak a felhasználási célban különböznek. Ezek könnyen összetéveszthetők még részletes adatok alapján is, kivonatos adatok alapján pedig nem is lehet őket megkülönböztetni.

Az aláíró interfészkomponenssel kapcsolatos legnyilvánvalóbb és legegyszerűbben teljesíthető követelmény a következő:

- Az aláírás végrehajtása előtt az aláíró jóváhagyását kell kérni az aláíráshoz.
  Ez azért szükséges, mert meg kell akadályozni azt, hogy az aláírás az aláíró akarata ellenére jöjjön létre. A jóváhagyást *(mely különbözik az aláíró aláíró-eszköz felé történő hitelesítésétől)*, minden egyes aláírás előtt ki kell kérni. Célszerű amolyan „utolsó figyelmeztetést" alkalmazni, mivel az aláíró nem biztos, hogy tisztában van vele, melyik az utolsó lépés, ami után az aláírás létrejön.

Az aláíróhitelesítő komponenssel kapcsolatban hasonlóan evidens ez a követelmény:

- Amennyiben az aláíró meghatározott számú próbálkozás után sem képes az aláíráslétrehozó eszköz aktiválási adatának helyes megadására, figyelmeztetni kell az aláírót, és meg kell akadályozni a további próbálkozásokat.
  A többszöri hibás kísérlet az aktiválási adat megadására próbálkozásos támadásra utal. Célszerű, ha az engedélyezett kísérletek száma konfigurálható a helyi előírásoknak és egyéni igényeknek megfelelő módon *(de ha ilyen konfigurációra mód van, az nem lehet hozzáférhető egy illetéktelen behatoló számára)*. Az engedélyezett kísérletek túllépése esetén az eszközt blokkolni kell, ami után csak egy másik kóddal *(PUK)* hozható újra normál állapotba.

Az aláírandó adatformattálóval szemben talán ez az egyetlen elvárás:

- Az aláíró által választott aláírási szabályzatnak megfelelően kell összeállítani a formattált aláírandó adatot. Amennyiben nem így történik az számos félreértéshez, könnyen az aláírás ellenőrzésének teljes meghiúsulásához vezethet.

A lenyomatoló komponenssel szemben számos kriptográfiai és formátum-megfelelőségi követelmény létezik. Egy ilyen az alábbi:

- Csak azok a lenyomatoló algoritmusok használhatók a komponensben, melyek a minősített elektronikus aláírások számára elfogadottak. Nem megfelelő biztonságú lenyomatoló algoritmus felhasználása biztonsági problémákat jelent.
  A hazánkban e célra elfogadott lenyomatoló algoritmusok listája a 2/2002. MeHVM irányelv 1-es számú mellékletében találhatók meg.

A BALE-kommunikátorral szembeni elvárások közül az alábbi egy elsősorban funkcionális kívánalom:

- Biztosítani kell annak lehetőségét, hogy az aláíró a BALE-n tárolt alkalmazások, aláíráslétrehozó adatok és egyéb aláírásjellemzők közül az általa felhasználni kívántat választhassa ki. Az alkalmazások közül csak a relevánsak kiválasztását szabad engedélyezni, a kiválasztott tanúsítványnak megfelelő aláíráslétrehozó adat használatát biztosítani kell, s az aláírásjellemzőkkel kapcsolatos szabályok kezeléséről is gondoskodni kell. A követelmény olyan esetben releváns, mikor a BALE-n több aláíráslétrehozó adat, és ezekkel összefüggésben több aláírásjellemző is tárolódik, illetve amennyiben multiapplikációs BALE-ról van szó. Ilyen esetben egy nem megfelelő, vagy az aláíró által felhasználni nem kívánt alkalmazás vagy aláíráslétrehozó adat kiválasztása nem megfelelő aláírás előállítását és egyéb hibákat eredményezhet.

Az aláírói dokumentumösszeállítóval kapcsolatban nincsenek követelmények explicit megfogalmazva, de az általa előállított dokumentumokon és azok kezelésén keresztül igen. Így áttételesen követelmény, hogy olyan formátumú dokumentumokat legyenek képesek előállítani, amit az alkalmazás aláírói dokumentum megjelenítő komponense képes az aláíró számára az eredeti formájában megjeleníteni. Az előállított dokumentum megfelelőségének tesztelése az aláírói dokumentum- és aláírásjellemzők megjelenítő komponensek feladata.

Az aláírt adathalmazösszeállítóval kapcsolatos alábbi követelmény nyilvánvaló, de nem könnyen teljesíthető:

- Gondoskodni kell arról, hogy az aláírt adathalmazban lévő minősített elektronikus aláírás érvényessége a szintén az aláírt adathalmazban szereplő adatok alapján megállapítható legyen.

Az aláírt adathalmaz formátuma nem feltétlen képezi le pontosan a formattált aláírandó adatot. Ezért az aláírt adathalmaznak tartalmazni kell az aláírás elkészítésében szerepet kapó összes adatot, vagy az ezekre történő egyértelmű utalást, valamint a formattált aláírandó adat típusát, hogy az aláírásellenőrző alkalmazás képes legyen azt az aláíráslétrehozó alkalmazáshoz hasonlóan összeállítani.

Az Aláírásnaplózó komponenssel szembeni következő pont ajánlásnak tekinthető:

- Lehetőség szerint lehetővé kell tenni aláírónak, hogy az aláíráslétrehozó eszköz felhasználását nyomon tudja követni.
  Ez naplózási és naplómegtekintési funkciók biztosításán keresztül történhet meg.

A szolgáltatói együttműködő komponenssel szembeni legfőbb elvárás, hogy:

- Az alkalmazásnak képesnek kell lenni tanúsítványok szolgáltatótól történő letöltésére és hitelességük, valamint érvényességük ellenőrzésére.

A BALE-tulajdonos kijelzővel szembeni egyetlen követelmény az alábbi:

- Gondoskodni kell a BALE-tulajdonos azonosító kijelzéséről, amennyiben ilyen adat rendelkezésre áll.
  Ez azért szükséges, hogy az aláíró ellenőrizhesse, hogy valóban a saját eszközét használja. A felirat nélküli eszközöket *(pl. USB tokenek)* könnyű elcserélni egy cégen belül, ahol mindenkinek ugyanolyan kütyüje van.

Almási János
CIO
Almasi-Janos@dbrt.hu

Terminológia

Minősített elektronikus aláírás
Olyan fokozott biztonságú elektronikus aláírás, amely biztonságos aláíráslétrehozó eszközzel készült, és minősített tanúsítványon alapul.

Fokozott biztonságú elektronikus aláírás
Elektronikus aláírás, amely alkalmas az aláíró azonosítására és egyedülállóan hozzá köthető; valamint olyan eszközökkel hozták létre, amelyek kizárólag az aláíró befolyása alatt állnak; és a dokumentum tartalmához technikailag olyan módon kapcsolódik, hogy minden – az aláírás elhelyezését követően az iraton, illetve dokumentumon tett – módosítás érzékelhető.

Elektronikus aláírás szolgáltatások
A jogszabályok a következő elektronikus aláírás szolgáltatásokat definiálják:

- (elektronikus aláírás) hitelesítésszolgáltatás
- időbélyegzés
- aláíráslétrehozó eszközön az aláíráslétrehozó adat elhelyezése

Hitelesítésszolgáltatás
A hitelesítésszolgáltatás keretében a hitelesítésszolgáltató azonosítja az igénylő személyét, tanúsítványt bocsát ki, nyilvántartásokat vezet, fogadja a tanúsítványokkal kapcsolatos változások adatait, valamint nyilvánosságra hozza a tanúsítványhoz tartozó szabályzatokat, az aláírás-ellenőrző adatokat és a tanúsítvány aktuális állapotára (különösen az esetleges visszavonására) vonatkozó információkat.

Megbízható útvonal
Az aláíráslétrehozó rendszer hardver és szoftver alkotóelemein áthaladó útvonal, mely garantálja minden rajta keresztül továbbított adat integritását, hitelességét és bizalmasságát. Az egyes interfészkomponensekhez szükséges, mint az aláírói interfész és a BALE-kommunikátor.

„Azt írsz alá, amit látsz"
Elv, melynek érvényre jutása garantálja, hogy az aláíró valóban az általa látott aláírandó dokumentumot írja alá.

„Azt lett aláírva, amit látsz"
Elv, melynek érvényre jutása garantálja, hogy az ellenőrző valóban azt a dokumentumot látja, ami alá lett írva, abban a formában, ahogy az aláíró is látta.

Elliptikus görbe kriptorendszerek II. rész

Elliptic Curve Cryptography – algoritmusok és a gyakorlat

Egy jó ideje egyre több helyen találkozhatunk az ECC betűhármassal, mint egy kriptorendszer jelölésével. A cikk első részében áttekintettük e viszonylag fiatal kriptorendszer elvi alapjait, most jöjjön néhány gyakorlati példa és algoritmus!

Remélem az első rész senkit nem rettentett el az ECC-től és az elmúlt egy hónap mindenkinek elég volt ahhoz, hogy a kissé száraz előkészítést magáévá tegye. Lássunk néhány algoritmust, melyek rettentően hasonlítanak néhány meglévőre...

Titkosítás és aláírás az elliptikus görbékkel

Az ECDLP-n alapuló rendszerek többsége aláíró *(például ECDSA)* vagy kulcscserélő *(például ECDH)* rendszer, mert gyors titkosításra ez a módszer is alkalmatlan, hasonlóan a többi nyíltkulcsos algoritmushoz. A továbbiakban eme algoritmusokkal ismerkedhetünk meg. *(A folytatás jobb megértéséhez ajánlott ismeretek: Diffie-Hellman kulcscsere, ElGamal titkosítás valamint az üzenetpecsét algoritmusok ismerete – legalább nagyvonalakban. Ezek nélkül is érthető lesz a folytatás, csak nehezebb lesz párhuzamot találni a már meglévő algoritmusok és az EC-alapú algoritmusok között...)*

ECDH – Elliptic Curve Diffie-Hellman kulcscsere

Az eredeti Diffie-Hellman a szimmetrikus titkosító rendszerek kulcsmegosztási problémáját oldatta meg. Hogyan is? A két résztvevő ugyanazokat a műveleteket végezte el egyező nyilvános és különböző titkos paraméterekkel, de azonos eredményt kaptak, melyet kulcsként használhattak. Az ECDH is ugyanígy működik, csak nem moduláris hatványozást használ, hanem a korábban megismert EC-műveleteket.

Alice és Bob megegyeznek egy **E** görbében és egy **G** pontban, utóbbit bázispontnak hívjuk. A továbbiakban eme paramétereket nyilvános rendszerparamétereknek tekintjük. Alice választ egy véletlen számot, *(amely kisebb, mint a **G** pont rendje)* és ugyanígy tesz Bob is: Alice száma legyen **a**, Bobé legyen **b**. Mindketten titokban tartják választásukat. A kulcscsere következő lépésében Alice kiszámolja **a*G** pontot, melyet elküld Bobnak, aki Alice műveletéhez hasonlóan kiszámolja **b*G** pontot és elküldi Alicenak. Végül Alice a Bobtól kapott **b*G**-t megszorozza a-val, így megkapja **a*b*G** pontot, valamint Bob az Alice-tól kapott **a*G** pontot szorozza meg saját titkos **b** számával, és eredményül ő is az **a*b*G** pontot kapja.

A közös pont valamely tulajdonsága *(például x vagy y koordinátája vagy éppen x + y)* használható kulcsként. A kíváncsi Eve-nek az abG pontot kellene kiszámolnia, de csak G, aG és bG pontokat ismeri, magukat a titkos a és b számokat nem. Az elliptikus Diffie-Hellman működését és lépéseit az alábbi egyszerű számpélda alapján követhetjük:

```
Nyilvános paraméterek:
  E: y² = x³ + 5x + 8 mod 23   és   G(8,10)
Titkos paraméterek:
  a=7 (Alice titkos száma)
  b=3 (Bob titkos száma)
Kulcselőkészítés:
  Alice számol:          Bob számol:
  1G(8,10)               1G(8,10)
  2G(13,4)               2G(3,4)
  3G(20,9)               3G(20,9)
  4G(22,18)
  5G(6,1)
  6G(2,18)
  7G(7,15)
  aG = (7,15)            bG = (20,9)
Kommunikáció:
  Kicserélik eredményeiket, aG-t megkapja Bob,
  bG-t pedig Alice
Kulcsegyeztetés:
  1bG (20,9)             1aG (7,15)
  2bG (12,18)            2aG (13,19)
  3bG (7,8)              3aG (6,1)
  4bG (22,5)
  5bG (8,13)
  6bG (13,4)
  7bG (6,1)
  abG (6,1)              baG (6,1)
```

ECElGamal – Elliptic Curve ElGamal titkosítás

Ahogy az eredeti ElGamal titkosítás a Diffie-Hellman algoritmus problémáján alapul, úgy építhető fel az elliptikus ElGamal is az ECDH-ra:

1. Alice és Bob választ egy **E** görbét és egy **G** bázispontot.
2. Mindketten választanak egy-egy véletlen **a** és **b** számot, mint titkos kulcsot.
3. Alice elküldi az **a*G** pontot, mint nyilvános kulcsot Bobnak.
4. Bob elküldi a **b*G** pontot, mint nyilvános kulcsot Alicenak.
5. Ha Alice üzenni akar Bobnak, az üzenetet leképzi a görbe egy vagy több **M** pontjára és generál egy véletlen **k** számot, mint viszonykulcsot. Elküldi Bobnak a *(kG, M+k(bG))* üzenetpárost.
6. Bob a következőképpen olvassa el az üzenetet: a kapott küldemény első felét megszorozza saját titkos **b** számával, így **bkG**-t kap, amit egyszerűen kivon a küldemény második feléből.

Eve támadásának feltétele az lenne, ha Bob átküldött nyilvános kulcsából ki tudná számolni **b** értékét vagy a titkosított

üzenet első részéből **k** számot. Jelenlegi ismereteink szerint egyikre sem képes elfogadható időn belül.

ECDSA – Elliptic Curve Digital Signature Algorithm

A következő algoritmus a FIPS 186-2-ben leírt DSA-hoz hasonlóan működik, hasonlóak a végzett műveletek, lépések is. Ahhoz, hogy Alice egy **M** üzenetet aláírva el tudjon küldeni, a következő paraméterek és eszközök szükségesek:

1. egy elliptikus görbe mod q felett *(nyilvános paraméter)*
2. egy **G** bázispont, melynek rendje n *(nyilvános paraméter, $n \geq 160$ bit)*
3. egy véletlen d szám *(mely kisebb, mint n-1)* és egy **Q**=dG pont. Alice kulcspárja *(d,Q)*, ahol d a titkos és **Q** a nyilvános kulcsa.

Az ECDSA aláíró algoritmusa

1. Alice választ egy **k** számot 1 és n-1 között
2. Kiszámolja **kG**=(x_1, y_1) pontot és **r**=x_1 mod n. Ha a pont x koordinátája zérus *(x_1=0)*, akkor új **k** számot választ. A pont x koordinátája lesz az aláírás egyik komponense, ezért jelöltük meg külön egy r betűvel.
3. Kiszámolja **k** multiplikatív inverzét n-re *(vagyis k^{-1} mod n értékét)*.
4. Kiszámolja a küldendő üzenet pecsétjét, melyre a szabvány az SHA-1 algoritmust ajánlja. Legyen hát **e** = SHA-1*(M)*!
5. Az aláírás másik alkotóeleme a következő kifejezés: **s**=k^{-1}*(e + dr)* mod n. Abban a szerencsétlen *(és meglehetősen ritka)* esetben, ha **s**=0, akkor az egész algoritmust elölről kell kezdeni. Itt azt is láthatjuk, hogy a 2. lépésben miért nem lehet **r**=0, mert az aláírás nem tartalmazná a titkos kulcsot!
6. Az **M** üzenethez és Alice-hoz tartozó aláírás: az ***(r,s)*** értékpáros.

Az ECDSA ellenőrző algoritmusa

Feltételezzük, hogy Bob, mint az aláírás ellenőrzője rendelkezik a hitelesített*(!)* rendszerparaméterekkel és Alice hiteles nyilvános kulcsával. Bob a következő lépések végrehajtásával ellenőrizheti egy aláírás helyességét:

1. Ellenőrzi, hogy az ***(r,s)*** egész számok megfelelő intervallumban vannak-e? *(Kisebbek-e n-1-nél?)*
2. Kiszámolja a kapott üzenet pecsétjét. Ehhez ugyanazt az algoritmust kell használnia, amit Alice használt: **e** = SHA-1*(M)*.
3. Kiszámolja s multiplikatív inverzét n-re: w = s^{-1} mod n. Ezért nem hagyhattuk az aláírás 5. lépésében, hogy s=0 legyen: nem létezne inverze!
4. Kiszámolja a következő részeredményeket: u_1=ew mod n és u_2=rw mod n.
5. Végül kiszámolja a $\mathbf{P}(x_1,y_1) = u_1\mathbf{G} + u_2\mathbf{Q}$ elliptikus pontot. Ha **P=O**, akkor biztosan nem jó az aláírás, egyébként legyen v = x_1!
6. Bob az aláírást csak akkor fogadja el, ha v = r .

Miért helyes az ellenőrzés?

Az aláírás-ellenőrzés helyességéhez az kell belátnunk, hogy az 5. pontban kiszámolt **P** pont nem más, mint az Alice által kiszámolt **kG** pont *(aláírási folyamat második lépése)*. Alice aláírásának egyik része a következő kifejezés:

```
s = k^-1(e + dr) mod n
```

Ha az egyenlet mindkét oldalát megszorozzuk s^{-1}k szorzattal, akkor azt kapjuk, hogy

```
k  s^-1(e+dr)
   s^-1e + s^-1dr
   we + wdr
   u1 + u2d (mod n)
```

Már csak egy lépés választ el attól, hogy a Bob által kiszámolt **P** pontra belássuk állításunkat:

```
P(x1,y1) = u1G + u2Q =
         = u1G + u2dG =
         = (u1 + u2d)G = kG
```

Vagyis, ha Bob a számítás eredményeképpen Alice véletlen pontját kapja vissza, azok x koordinátáinak is meg kell egyezniük. Ha Bob egy másik pontot kap eredményül, az aláírás nem fogadható el.

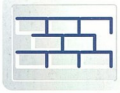
Pontok, görbék előállítása

Egy-egy ECC-rendszerhez szükség van egy **E** görbére, egy **G** bázispontra, véletlen pontokra stb. Az alábbiakban ezek előállítására láthatunk módszereket. Sajnos a görbék és pontok választásánál sok olyan tulajdonságra kell figyelni, amelyekről már volt szó, vagy eddig nem tértem ki rájuk és nem is fogok, azok komplexitása miatt. Ha ezen ismeretek hiányában kívánunk üzleti célú biztonságos implementációt készíteni, a szabványokban javasolt görbékből és bázispontokból válasszunk! Ne feledjük: ezek egyébként is nyilvános paraméterek! Például a **[sec]** linken elérhető ajánlásokban 113 bittől 571 bitig találunk paramétereket, **[fedstd]** ajánlásban pedig a szabványosnak tekintett bitméretekre: 112, 128, 160, 192, 224, 256, 384, 521 bit. Úgy vélem, hogy az ECC alapteremtő magyarázatához és megértéséhez nem szükségesek a most „elfelejtett" tulajdonságok, akit pedig érdekel, az Interneten is tengernyi irodalmat találhat. *(Személy szerint az IEEE P1363 szabványt ajánlom, én ebben találtam a legtömörebb, legegyszerűbb és „legimplementáció-barátabb" leírásokat.)* Kérem a Kedves Olvasót, hogy nézze el ezt a hiányosságot nekem, de csak és kizárólag az ECC matematikai hátteréről több könyvet lehetne írni. És még néhányat a kriptográfiai alkalmazásukról, gyakorlati implementációikról... Csak két példa „elrettentésül":

1. Minden eddig leírt definíciót, szabályt és algoritmust még legalább kétszer le lehetne írni, mert az elliptikus görbéket nem csak a természetes számok *(mod p)* felett lehet értelmezni, hanem polinom-algebra alapján is, amelynek legalább kétféle ábrázolásmódja ismeretes. *(Ilyenkor a modulus nem egy prímszám, mint eddig, hanem egy kettőhatvány. Szoftvermegvalósításban általában az előbbi, hardveres megoldásban az utóbbi a gyorsabb.)*
2. A kedvelt és szemléletes Descartes-féle koordinátarendszer mellett az elliptikus görbe pontjait a projektív síkon is szokták ábrázolni. Ekkor a végtelenben lévő **O** pont az origóba kerül.

Görbekészítés

A görbék készítésére egyébként legalább háromféle módszer van. Az egyik speciális görbékkel dolgozik, amelyek együtt-



hatói bizonyos szempontoknak megfelelnek: optimális hardvermegvalósítást tesznek lehetővé, vagy különlegesen ellenállóvá teszik a görbét valamelyik támadási forma ellen, stb. A második szerint a görbéket meghatározó együtthatókat véletlenszerűen választjuk meg:

```
Válasszunk egy prímszámot:
  p = 4294967861 (2^32+565)
Majd egy véletlen a paramétert:
  a = 1234567890
És egy véletlen b paramétert:
  b = 0987654321
Ellenőrzés következik:
  4a^3 + 27b^2 mod p = 75267055134940680039174941O7
mod 4294967861 = 1240368550 ≠ 0 → OK!
A kész görbénk egyenlete:
  y^2 ≡ x^3 + 1234567890x + 987654321 (mod 429467861)
```

A harmadik módszer nem véletlen számokat használ, hanem egy kezdőértékből *(seed)* számított SHA-1 érték alapján állítja elő az együtthatókat. Az IEEE P1363 és a NIST egyaránt ezt az eljárást ajánlja az ilyen típusú görbék *(pseudo-random curves)* konstruálásához **[fedstd]**. Az álvéletlen megoldás előnye, hogy reprodukálható és így ellenőrizhető, hogy egy görbe ezzel a módszerrel készült-e vagy sem.

Pontkészítés

Ha már van egy görbénk, azon egy véletlen pontot is kereshetünk. Példaképpen a véletlen számokból készített görbénken keresünk egy pontot:

```
y^2 = x^3+1234567890x+987654321 (mod 429467861)
x = 147896325 (véletlenszerűen választott)
y^2 = 370713451 (mod 4294967861)
y = ?
```

Hát, izé… ennek nincs megoldása, vagyis nincs olyan szám, amelynek négyzetét a modulussal elosztva 370713451-et kapnánk maradékul. Próbáljuk meg újra egy másik x értékkel!

```
x = 225589
y^2 = 376919525 (mod 4294967861)
y = 57372704
```

Na, ezzel megvolnánk: **P***(225589, 57372704)*! E pontnak van még néhány tulajdonsága, amit jó lenne tudni *(például generátor-e? Ha nem, mennyi a rendje?)*, de ezekkel most nem foglalkozunk *(lásd korábban, hogy miért nem)*.

Üzenet leképzése egy pontra és vissza

Néhány kriptográfiai algoritmus tartalmaz egy olyan lépést, amikor az üzenetet le kell képezni egy olyan alakra, amit az adott algoritmus kezelni tud. Esetünkben ez azt jelenti, hogy egy adott **E** görbe alkalmazása mellett Alice **P** ponttá tudja alakítani az m üzenetet és Bob egy megoldott pontból ki tudja venni az üzenetet. *(Itt jegyzem meg: előfordulhat, hogy csak a pont x koordinátája közlekedik teljes egészében a kommunikációban. Az y-nak csak legnagyobb helyértékű bitje kíséri az x-et, mondván, az y kiszámolható. Ebben az esetben a pontot tömörített pontnak (compressed point) hívja az ANSI9.62, az IEEE P1363 és a SEC is. A három forrás legnagyobb különbsége, hogy a SEC csak használja a fogalmat, a többiek el is magyarázzák.)*

A példához ismét a korábban készített görbénket fogjuk használni. Első próbálkozásunkkal alakítsuk ponttá a „Jó!" karaktersorozatot! Ehhez először számmá kell alakítani: a karakterek ASCII kódját hexa formában egymás mellé írjuk és egyetlen hexadecimális számként értelmezzük. Más módszer is használhatunk, a lényeg, hogy:

- kölcsönösen egyértelmű megfeleltetés legyen,
- a blokkméretnek kisebbnek kell lennie, mint a modulus hossza!

És ezután mi sem egyszerűbb, ez legyen az üzenetet képviselő **P** pont x koordinátája, csak ki kell számolni a hozzátartozó y-t! Lássunk neki!

```
m = „Jó!" = 0x4A 0xF3 0x21 = 0x4AF321 = 4911905
P(m,y) = (4911905, ?)
y^2 = x^3 + 1234567890x + 987654321 (mod 429467861)
y^2 = 43578828
y = 165701469
P(m,y) =  (4911905, 165701469)
```

Természetesen felvetődhet a kérdés, hogy mi van akkor, ha az üzenet alapján nem létezik pont? A pontgenerálás első próbálkozása is ezért volt sikertelen. Mi a teendő akkor, ha y2-nek nincs megoldása?

```
m = „Nem" = 0x4E 0x65 0x6D = 0x4E656D = 5137773
P(m,y) = (5137773, ?)
y^2 = x^3 + 1234567890x + 987654321 (mod 429467861)
y^2 = 109210672
y = nincs megoldása
```

Az ilyen esetekre felkészült alkalmazások nem tisztán az m üzenetet tekintik az x koordinátának, hanem az x koordináta felső bitjeibe helyezik el azt, majd az alsó bitekkel addig „szórakoznak", amíg eredményre nem jutnak: **P***(m * eltolás + valami, y)*. Ilyenkor az üzenetet kibányászni a *(P_x / eltolás)* egészrészeként lehet. Példánkban a modulus 29 bites, az eltolás legyen 6 bit, így 23 bites üzeneteket tudunk továbbítani. Próbáljuk meg még egyszer a „Nem" szót ponttá alakítani, az eltolás használatával *(valami=10, ha nem jutunk eredményre, majd választunk másikat…)*:

```
m = „Nem" = 0x4E 0x65 0x6D = 0x4E656D = 5137773
P(m*2^6+10,y) =  (328817482, ?)
y^2 = x^3+1234567890x+987654321 (mod 429467861)
y^2 = 275000646
y = 125641602
P(328817482, 125641602)
```

Ezt a pontot már Alice tetszőleges módon titkosíthatja és elküldheti Bobnak. Bob a dekódolás után szerencsés esetben ezt a pontot fogja visszakapni. Megragadja a pont x koordinátáját, elosztja 2^6-nal és az eredmény egészrésze: 5137773 ! Voilá!

Tényleg biztonságban vagyunk?

A kérdés megválaszolásához a bevezetőben szereplő gondolatokat kell továbbfűznünk az eddigi törési kísérletek és tapasztalatok fényében.

Certicom challenges

Az RSA Inc.-hez hasonlóan a Certicom Corporation **[certicom]** is írt ki törési versenyeket, melyek egy része mára megoldott, másik része még megoldásra vár **[certchall]**. A feladatok 1999 óta – a Certicom szándéka szerint – azt kívánják bizonyítani, hogy az ECC erősebb, mint az RSA- vagy a DLP-probléma. Másrészt marketingfogás, amely megpróbálja a potenciális ipari felhasználók, fejlesztők és a kutatók figyel-

mét az ECDLP felé terelni. A versenykiírásban mintegy 20 nyilvános kulcs szerepel, a hozzájuk tartozó rendszerparaméterekkel együtt [curlist]. A feladat: meg kell keresni a titkos kulcsot!

A Certicom az alábbi három csoportra osztotta a feladványokat *(zárójelben a Certicom által becsült megoldási idők, néhány ezer együttműködő gép esetére)*:

Excercises:	79 bites	*(néhány óra)*
	89 bites	*(néhány nap)*
	97 bites	*(néhány hét)*
Level I:	109 bites	*(néhány hónap)*
	131 bites	*(néhány hónapnál sokkal több)*
Level II:	163 bites	*(jelenleg megoldhatatlan feladatok)*
	191 bites	
	239 bites	
	359 bites	

Eddigi megoldások néhány technikai adatát tartalmazza a következő felsorolás. *(Érdekes, hogy a különböző források kis mértékben ellentmondóak egymásnak, akárcsak az RSA törési versenyek esetében...)*

2002. November 6. – ECCp-109 Challenge megoldása *(prím modulus)*: 10300 résztvevő, 10000 számítógép, 1,5 év időtartam;

2000. Április 17. – ECC2K-108 Challenge megoldása *(kettőhatvány modulus)*: 1300 résztvevő, 9500 számítógép, 4 hónap időtartam;

1999. Szeptember 28. – A 97-bites ECC Challenge megoldása: 200 résztvevő, 740 számítógép, 16000 MIPS-év számításigény. Mindez körülbelül fele az ötször hosszabb RSA-512 feltöréséhez használt teljesítménynek.

Pollard-ρ algoritmusa

Napjaink legjobbnak tartott algoritmusa a Pollard-ρ algoritmus *(Pollard-ró)*. Az eljárás kis módosítással teljes mértékben párhuzamosítható, így ha 10 processzor áll rendelkezésre, 10-szer gyorsabban jut eredményre. Ha csak egy processzorunk van, $0.5*\sqrt{(\pi*\rho)}$ EC-összeadás *(ahol ρ a modulus)* kell a végrehajtásához, ha több, akkor ez a sok számítás megoszlik köztük. Akármilyen jó is az algoritmus, tetemes számításigénye van:

ρ mérete	$0.5*\sqrt{(\pi*\rho)}$	MIPS-év
97 bit	2^{49}	$1{,}6\times10^{4}$
160 bit	2^{80}	$8{,}5\times10^{11}$
186 bit	2^{93}	$7{,}0\times10^{15}$
234 bit	2^{117}	$1{,}2\times10^{23}$
354 bit	2^{177}	$1{,}3\times10^{41}$
426 bit	2^{213}	$9{,}2\times10^{51}$

Összehasonlításul a faktorizálás becsült számításigénye a szokásos modulusok méretére:

modulus mérete	MIPS év
512 bit	3×10^{4}
768 bit	2×10^{8}
1024 bit	3×10^{11}
1280 bit	1×10^{14}
1536 bit	3×10^{16}
2048 bit	3×10^{20}

(1 MIPS-év az a számításigény, ami 1 darab 1 MIPS teljesítményű számítógéppel 1 év alatt teljesíthető.)

Az algoritmus további részleteitől és hátterétől most nagyvonalúan tekintsünk el, jelentősen túlmutat a cikk célkitűzésein.

Válasz a kérdésre: nem tudjuk!

Napjaink minden széles körben elterjedt kulcscserére, titkosításra vagy digitális aláírásra használt PKI algoritmusa a faktorizálás vagy a diszkrét logaritmus problémáján nyugszik. A két probléma hasonlít egymáshoz, amit az is jól jelez, hogy a legjobb faktorizáló algoritmusok *(bizonyos feltételek teljesülése esetén)* felhasználhatók a DLP-problémák megoldásában is. Némi egyszerűsítéssel azt is mondhatjuk, hogy egyező kulcsméret mellett egyező biztonságot nyújtanak.

Sajnos a DLP és az ECDLP már nem hasonlít ennyire egymásra, a viszonylag jó és újabb DLP-megoldó algoritmusok *(„index kalkulus")* egyszerűen nem használhatók ECDLP esetére: ott meg kell elégedni a régebbi módszerekkel *(például Pollard-ρ)*. Ebből az is következik, hogy elégséges, ha a kulcsok e régi és lassú „trükköknek" ellenállnak, tehát rövidebbek is lehetnek. Jelentősen rövidebbek. A minimálisan ajánlott kulcsméret ma 1024 bit az RSA esetében, 163 bit az ECC-rendszerekhez.

Semmi sem garantálja azonban, hogy ez holnap is így lesz. Semmi sem garantálja, hogy a közeljövőben valaki nem publikál egy olyan eljárást, amely valóban jó megoldást nyújt az ECDLP-re. Nem tudjuk, hogy elvileg sem működnek a DLP jó algoritmusai vagy csak a mi ismereteink a hiányosak. Igazság szerint az ECDLP-re ma is léteznek jó algoritmusok, de ezek mindegyike kizárólag valamilyen speciális tulajdonságú görbére és nem általánosan alkalmazható. Jelenlegi tudásunk szerint mindenestre az ECDLP alapú kriptorendszerek jobbak – gyorsabbak és biztonságosabbak –, mint az IFP-n vagy a DLP-n alapuló kriptorendszerek **[menezes]**.

Virasztó Tamás
wacher@westel900.net

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

Objektumorientált alkalmazásfejlesztés

Az UML

Sok szó esett már az UML-ről, és talán még mindig vannak, akik számára nem egyértelmű, mit is takar ez a három betű tulajdonképpen. Számukra szeretnék egy kis segítséget nyújtani, hogy tisztább legyen a kép, és ne rémüljenek meg, ha egy osztálydiagram feltűnik valahol.

Az elmúlt hónapban néhány szóban igyekeztem bemutatni az objektumorientált világot, az OO fejlesztés szükségességét és létjogosultságát. Az ott felsorolt fogalmak, rövidítések nem választhatók el szervesen egymástól, most mégis igyekszem egyet kiemelni közülük – ez pedig az UML lesz.

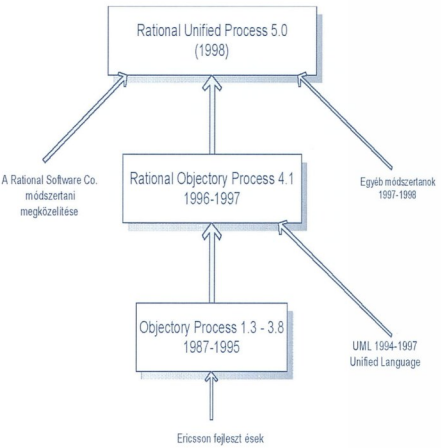
Láttuk, hogy a szoftverkrízis megoldásaként az objektumorientált programozás megjelenése szolgált. Ez azonban újfajta szemléletet jelentett, így újabb és újabb tervezési módszerekre volt szükség. Felismerték, hogy nagy és bonyolult rendszereket csak hatékony módszerekkel lehet tervezni és fejleszteni. Olyan, a tervezés és fejlesztés során egységesített technikákra van tehát szükség, amelyek segítségével a folyamat nemcsak gyorsabb, de átláthatóbb, kezelhetőbb, ezáltal sokkal eredményesebb is lesz.

A '70-es évek közepétől így folyamatosak voltak azok az erőfeszítések, amelyek alternatív elemzési és tervezési módszereket kerestek. Számos megoldás született, többségük azonban nem volt alkalmas arra, hogy a gyakorlatban is elterjedjen.

Az áttörés a '90-es évek elején történt, amikor olyan új generációs, a szoftver teljes életciklusát átfogó, hatékony módszertanok láttak napvilágot, amelyek számos projektben eredményesnek bizonyultak. Néhány példa ezek közül: OMT, Booch'94, Coad-Yourdon, Fusion stb. A sokféle módszertan közül nehéz volt választani, hiszen mindegyiknek megvolt a maga erőssége és gyengesége egyaránt. Egyre sürgetőbbé vált az egységesítés. A különféle módszerek különféle jelölésmódot alkalmaztak, ami igencsak megnehezítette a fejlesztők dolgát, akik projektenként mást voltak kénytelenek megtanulni. Ugyancsak nehézkes volt így a fejlesztők és felhasználók közötti kommunikáció is.

Ezen igényeket és hiányosságokat felismerve tűzte ki célul egy egységes módszertani és modellezési megoldás kidolgozását James Rumbaugh *(General Electric)*, Ivar Jacobson *(Objectory AB)* és Grady Booch *(Rational Software Co.)*, akik a korábbi módszerek vezető fejlesztői voltak. Korábbi, külön-külön megszerzett tapasztalataik alapján ki akarták szűrni azokat a hátulütőket, amelyekkel munkájuk során találkoztak, és az egységesítéstől egyfajta stabilitást vártak az OO tervezési-elemzési módszerek piacán.

Az UML-ig *(és a RUP-ig)* vezető út csaknem 10 évig tartott, folyamatát az alábbi, Booch-tól kölcsönzött ábra mutatja:



A RUP kifejlesztésének folyamata

Az egységesített módszertan tehát a RUP *(Rational Unified Process)* nevet kapta, fejlesztése 1998-ban ért véget. A módszertan az UML-t használja modellező nyelvként, és az általa definiált folyamat alapvetően iteratív jellegű: a tervezési/fejlesztési folyamat ismétlődő iterációkból áll, amelyek során a rendszer kidolgozottsága egyre finomabbá válik.

Most azonban nem célom ezt a folyamatot részleteiben bemutatni, „csupán" az UML-ről mint modellező nyelvről szeretnék írni néhány hasznos dolgot. Mindezt igyekszem úgy megtenni, hogy a kezdők számára is érthető legyen, az UML-ben járatos szakemberek pedig egyfajta áttekintést kaphatnak majd.

Az UML története

A '90-es évek elején már felmerült az igény egy vizuális modellező nyelv kialakítására, az érdemi munka azonban csak 1994-ben kezdődött, amikor Rumbaugh csatlakozott a Rational csapatához, és Booch-hal együtt kidolgozták az 1995-ben publikált Unified Method elnevezésű módszertant,

amelyet tulajdonképpen az UML *(Unified Modelling Language)* 0.8 verziójának tekinthetünk.
Az UM nyilvánossá tétele után csatlakozott a csoporthoz Jacobson, aki újabb előnyös elemek felvételét javasolta az UM-hez. Sok elem került tehát a régebbi módszertanokból az UM-be, ugyanakkor sok olyan újdonság is megjelent, amely korábban sehol nem volt megtalálható. Az újabb és újabb elemek hozzáadásával egyébként igen óvatosan bántak, hiszen nem akarták feleslegesen túlbonyolítani az új megoldást. 1996-ban jelent meg az UML 0.9, amelyet már ki is adtak különböző fejlesztő csapatoknak tesztelés céljából: használják ezt a modellező nyelvet, és tapasztalataikat összák meg, hogy azok felhasználhatók legyenek a későbbiekben. Ezek a kísérletek sikeresek voltak, és sok cég innen kezdve már kizárólag az UML-t használta *(és használja)*. Az OMG 1997 végén fogadta el az UML 1.1 verzióját, amelynek fejlesztésébe már különböző neves cégek is bekapcsolódtak *(IBM, Microsoft, HP, stb.)*.
A nyelv hiányosságai és problémái azonban – mint általában – most is a használatbavételt követően jelentkeztek igazán. Egy nyelvet azonban nem célszerű túl gyakran váltogatni, hiszen idő kell egyrészt annak elsajátításához, másrészt a használatát támogató CASE eszközök kifejlesztéséhez is. Eltérő vélemények vannak arról, hogy melyik irányba ildomos eltolni a hangsúlyt *(a folyamatos frissítést vagy a stabilitást tartsuk-e szem előtt)*. Rumbaugh szerint egy lehetséges és hatékony, kompromisszumos megoldás az, ha 4-5 évente felülvizsgáljuk és frissítjük nyelvet.
Az UML-lel kapcsolatos észrevételek elsősorban a kiterjeszthetőségre, a feltételek megfogalmazásának módjára, a nyelv struktúrájára és a diagramok közötti átjárhatóságra vonatkoztak. A tervek szerint 2001 végére készült volna el az új nyelv dokumentációja, és 2002 elején fogadták volna el a végleges verziót. Némi csúszással mindez nemrég történt csak meg, ezért cikksorozatomban elsősorban a „régi" UML elemeit mutatom be, és külön térek majd ki az UML 2.0 újdonságaira, a bekövetkezett módosításokra és változásokra.

Az UML elemei

Az UML-ről és a használatáról korábban már írtam egy cikksorozatot. Úgy gondolom azonban, hogy a téma kimeríthetetlen, és ezúttal igyekszem mélyebben belemenni egyes részletekbe, illetve megpróbálom áthelyezni a hangsúlyt az objektumorientált szemléletmódra, annak szükségességére. Most nem egy konkrét problémát mutatok be *(mint annak idején a biliárdjátékot)*, hanem kisebb, de lényegretörőbb példákon keresztül igyekszem szemléltetni az egyes elemek jelentőségét.

Egy UML modell alapvetően kétféle elemből állhat: egyrészt szöveges, másrészt vizuális részekből. Az előzőekre jó példa a különböző használati esetek felsorolása, leírása, az utóbbira jónéhány példadiagramot felsorakoztattam a korábbiakban.
Az objektumorientált világban elengedhetetlen az osztályokban történő gondolkodás, így először is járjuk körbe ezt a kérdéskört. Mint azt már korábban többször kifejtettem, az osztályok az emberi gondolkodást igyekeznek tükrözni, így első ránézésre értelmetlennek tűnhet nagy hangsúlyt fektetni rájuk. „Ha olyan közel áll a gondolkodásomhoz, minek annyit gondolkodni rajta?" – merülhet fel a kérdés. Itt most valóban nem pszichológiai fejtegetésekről lesz szó. Azt igyekezzük felderíteni, hogy az intuitíven meghatározott osztályok hogyan közelíthetők az optimális megoldáshoz, azaz ahhoz, ahogyan a leghatékonyabban leképezhetjük azt egy szoftverben.
A hatékonyság itt nemcsak azt jelenti, hogy a szoftver működése a lehető leggyorsabb legyen. Magával a kóddal szemben is támasztunk különböző követelményeket: könnyen olvasható, érthető, áttekinthető legyen, hiszen bármikor szükség lehet arra, hogy átadjuk azt egy másik kollégának, és egy idő után a saját kódunkban sem tudunk eligazodni, ha nem követjük ezeket az irányelveket.
Másik elvárás az, hogy a program könnyen módosítható, bővíthető legyen, azaz fel legyen készítve újabb modulok hozzáadására, illetve ha a meglévő modulokat módosítani kell, ne kelljen újraírni az egész rendszert, hanem a lehető legkisebb erőfeszítéssel ezt megtehessük.
Ehhez azonban jól átgondolt struktúrára, jól kialakított osztályokra van szükség.
Lássuk tehát, hogyan írhatunk le egy osztálystruktúrát. Természetes módon adódik, hogy alapvető elemei az osztályok, amelyek között különféle kapcsolatokat definiálhatunk. Erre különféle eszközök állnak rendelkezésünkre. A két legismertebb a Microsoft Visio *(amelyet én magam is használni fogok az elkövetkezőkben)*, és a Rational Rose *(talán nem meglepő, hogy a Rational rendelkezik ilyen eszközzel…)*.
Ha a Visiot elindítjuk, egy menüben választhatjuk ki, milyen célra szeretnénk aktuálisan használni. Válasszuk a Software csoportból az „UML Model Diagram" opciót. Ekkor a megnyíló ablakunk három részből áll: középen látható maga az UML modell, ami első indításkor természetesen üres, később töltjük fel tartalommal. Bal oldalon találjuk az egyes UML elemek felsorolását, amelyeket hozzáadhatunk modellünkhöz:

- Use Case
- Static Structure
- Statechart
- Sequence
- Deployment
- Component
- Collaboration
- Activity

Lássuk, mit is jelentenek ezek az elemek:

- **Use Case:** Használati esetek diagramjai definiálhatók. Az esetek többségében szükség van ezek szöveges leírására is. Később látunk rá példát.
- **Static Structure:** A rendszer statikus osztálystruktúráját írhatjuk le segítségével.
- **Statechart:** Az állapotátmenet-diagram rajzolásához ad eszközöket.
- **Sequence:** Különböző szekvenciadiagramok rajzolására.
- **Deployment:** A rendszer telepítésével, fizikai felépítésével kapcsolatos információk meghatározására.
- **Component:** A rendszer komponenseit rögzíthetjük.
- **Collaboration:** Együttműködési diagramok rajzolására.
- **Activity:** Akciódiagramokat definiálhatunk *(később ezekre is látunk példát)*.

Láthatjuk, hogy egy UML modell igen összetett is lehet. Persze a megismerés útján elindulva nem zúdul a nyakunkba rögtön az összes UML diagram, ezek fokozatos elsajátítása

sokkal hasznosabb lehet – persze ha közben tisztában vagyunk azzal, hogy teljesértékű modellt csakis ezek együttes használatával kaphatunk.

A fokozatosság elvét követve tehát térjünk rá az osztálydiagramokra, azaz a rendszer statikus struktúrájára. A diagramok megrajzolásához első lépésben azonosítani kell az adott problémát leíró osztályokat és ezek kapcsolatait. Első megközelítésben intuitív módon igyekszünk felvázolni a megoldást, vagyis azonosítjuk a rendszer egyes különálló moduljait, ezek felelősségét, és a közöttük fennálló kapcsolatokat. Például egy vállalati szoftver esetében szükség lehet számlázó, raktárkészlet-nyilvántartó, megrendelés-kezelő, adóbevallás-készítő, és egyéb modulokra. Ezek mindegyikének jól meghatározott feladata van, ezt részletesen kifejthetjük már a tervezés kezdeti szakaszában is.

Például:

```
Számlázómodul:
Feladata a különböző számlák kiállítása és nyomtatása. A számlán szerepelnie kell az aktuális jogszabályoknak eleget tevő adatoknak (lásd ÁFA-törvény X-edik paragrafus), valamint az eladáshoz kapcsolódó termékek illetve szolgáltatások mennyiségének, ÁFA-tartalmának, egységárának és együttes értékének.
A felhasználó egy űrlap kitöltésével állíthat ki újabb számlát. A számlán szereplő tételek mind egyedi azonosítóval ellátott termékek és szolgáltatások, melyeket egy külön erre a célra összeállított felületen vihetünk be a rendszerbe.
A számla nyomtatása a bevitt adatok alapján automatikusan történik a jóváhagyással egyidőben.
A nyomtatásnál fontos kritérium, hogy többoldalas számla esetében mindegyik lapon szerepelnie kell a fejlécnek, ami tartalmazza az eladó és a vevő adatait, valamit az aktuális dátumot és a számla sorszámát.
```

Előírhatunk olyan követelményeket is, amelyek az egyes modulok közötti kapcsolatokat határozzák meg:

```
A számlákon csak és kizárólag olyan termék szerepelhet, amiből kellő mennyiség áll rendelkezésre a raktáron. A számla kiállításával a raktáron lévő mennyiség a számlán szereplő mértékben csökken.
```

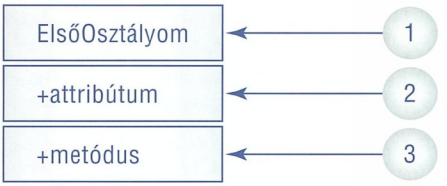
Így egy olyan rendszerváz áll rendelkezésünkre, amely alapján már elkezdődhet a konkrét megvalósítás tervezése. Hangsúlyozom, hogy ez a fázis még a megrendelő bevonásával történik, hiszen az ő intenzív közreműködése szükséges a rendszer alapkövetelményeinek meghatározásához.

Az osztályhierarchia meghatározásához alapul ezeket a fent meghatározott komponenseket használjuk, majd ezt a modellt finomítjuk. A számlázásnál maradva abból indulunk ki, hogy szükségünk lesz erre a modulra. Az előbb vázolt követelményekből már látszik is, hogy a számlázás több alfeladatból áll: számla kitöltése, tárolása, nyomtatása, stb. Vajon milyen osztályok valósítják meg ezeket a feladatokat, van-e szükség külön osztályokra minden egyes feladatkörhöz?

Első megközelítésben persze kiindulhatunk abból, hogy egyetlen osztály felel majd a számlázásért, s ennek lesznek különböző metódusai, amelyek a nyomtatást, tárolást, stb. végzik.

Jogosan merül fel az igény, hogy ha már osztályokról beszélünk, lássunk is egyet, ábrázoljuk, lássuk el különböző jellemzőkkel stb. Ám legyen.

Az osztály szimbóluma UML-ben egy téglalap, amelyet három részre osztunk az alábbiak szerint:

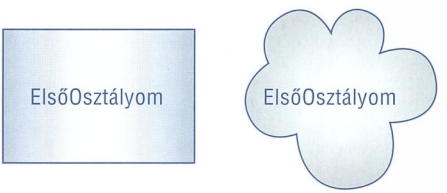


Osztály ábrázolása UML-ben

Az ábra egyes részei:

1. Az osztály neve
2. Az osztály attribútumainak listája
3. Az osztály metódusainak listája

Érdekességként említem meg, hogy az UML elődeiként élő egyes módszertanok más és más szimbólumokat használtak az osztályok jelölésére. Persze az osztályfogalom is eltért többé-kevésbé az UML-ben használatostól. Néhány példa:



Egy osztály ábrázolása az OMT és a Booch-féle szimbólumrendszerben

Az osztályt a neve egyértelműen azonosítja. Különböző megvalósításokban különbözőek az elnevezési tradíciók, ráadásul ezek cégenként, projektenként is eltérhetnek. Lehet, hogy az osztályneveket névterekbe soroljuk, ekkor az osztály a névtér:osztálynév páros határozza meg. Lehet, hogy a tradicionális nevezéktan azt írja elő, hogy valamilyen előtagot tegyünk az osztályunk neve elé *(például classMyFirstClass)*. Szokás az osztály nevében szereplő szavakat nagybetűvel kezdeni a könnyebb olvashatóság kedvéért. Általánosan is elmondható, hogy egy cégen illetve projekten belül célszerű egyfajta nevezéktant kialakítani, amelynek betartása mindenkire egyaránt kötelező. Így kommunikálni és együtt dolgozni is sokkal egyszerűbb, hatékonyabb, hiszen elkerülhetjük például azokat a szituációkat, amikor egy osztályt BélaOsztálya névvel illetnek az éles rendszerben *(ne tessék nevetni, találkoztam már ilyennel)*.

Persze ezek az észrevételek nemcsak az osztályokra érvényesek...

Az osztályok következő fontos tényezői az attribútumok. Ezek az adattagok az osztály működéséhez szükséges változók, szerepük különböző lehet. Elképzelhető, hogy az osztály belső működéséhez, „életéhez" szükséges egy új változó bevezetése, ekkor felesleges azt kifelé is publikálni. Lehet, hogy az adott változó a külvilág számára is hordoz információkat, ekkor láthatóvá kell tenni. Olyan eset is elképzelhető, amikor az elrejtés nem járható út, mert az osztályból leszármazó egyéb osztályoknak szüksége van az adott változóra *(a származtatásra és az öröklődésre később részletesen kitérek)*, viszont teljesen publikussá sem szeretnénk tenni. Természetesen ez is megoldható, pontos menetét később bemutatom. Előtte azonban lássuk az osztályok metódusait. Ezek azok a cselekvések, amire az osztály egy példánya képes. Ide tartozik az adott példány születése, megszűnése, és egyéb specifikus dolgok. A metódusokra is értelmezettek a fenti láthatósági feltételek, azaz itt is meghatározhatjuk, ki számára szeretnénk elérhetővé tenni ezeket.

Általános alapelv, hogy mindent a lehető legszűkebb körben tegyünk láthatóvá, vagyis aki elől csak tudjuk, rejtsük el. Ennek ára lehet, hogy például az attribútumokat privát használatúvá tesszük, de definiálunk hozzájuk elérő metódusokat: egyet a kiolvasására, egyet a beállítására *(amennyiben a logika szerint ez kívülről közvetlenül lehetséges)*.

Miért jó mindez? Képzeljük el azt a szélsőséges esetet, amikor minden attribútum *(szokásos még a paraméter elnevezés is)* és minden metódus elérhető mindenki számára. A többi osztályban létrehozzuk az adott osztály egy példányát, majd ha úgy gondoljuk, hogy valamely tagváltozójának értéke nem felel meg számunkra, rögtön nekiesünk, és már át is állítjuk azt, függetlenül attól, hogy logikailag esetleg ez nem megengedett, mert csak az osztály belsejében, egyéb paraméterek alapján történhetne értékadás.

Másik bökkenő, ha módosul az osztályunk felépítése: átnevezzük az adott tagváltozót, vagy egy újabb algoritmus használatával igyekszünk hatékonyabbá tenni egy-egy metódust. Ha minden mindenki számára elérhető, és a fent bemutatott módon, közvetlenül használunk mindent, millió helyen kényszerülhetünk átírni az alkalmazásunkat, mert nincs egy olyan egységes interfésze az osztályunknak, amely elfedné ezeket a belső dolgokat.

Lássunk egy példát erre is. Legyen adott egy olyan osztály, amely – múlt havi példámnál maradva – torták elkészítéséért felelős. A Torta osztály felépítése most némileg eltér a korábbitól:

Torta
+előkészítésiIdő : int
+pihentetésiIdő : int
+sütésiIdő : int
+krémKeverésiIdő : int
+összeállításiIdő : int
+szaummaElkészítésiIdő : int
+szummaidőSzámítás : int

A Torta osztály

Jól látható, hogy az osztálynak több idő-paramétere is van. Feltételezzük, hogy a szummaElkészítési idő az egyes idők összege, hiszen mondjuk egyetlen személy sürgölődik a konyhában, így nem párhuzamosíthatók a tevékenységek. Ezért létrehozunk egy szummaldőSzámítás() nevű metódust, amelynek belseje valahogy így néz ki:

```
public int előkészítésiIdő;
public int pihentetésiIdő;
public int sütésiIdő;
public int krémKeverésiIdő;
public int összeállításiIdő;
public int szummaElkészítésiIdő;

public int szummaIdőSzámítás()
{
       szummaElkészítésiIdő = előkészítésiIdő +
              pihentetésiIdő +
              sütésiIdő +
              krémKeverésiIdő +
              összeállításiIdő;
       return szummaElkészítésiIdő;
}
```

Ez mind szép és jó, működik is, van azonban egy apró bökkenő. Mi van akkor, ha a felhasználó az osztály példányosításakor *(vagyis amikor konkrét objektumot hoz létre belőle)* nem veszi figyelembe ezt, és a szummaElkészítési időnek is közvetlenül szeretne értéket adni? Valahogy így:

```
[1] Torta dobostorta = new Torta();
[2]
[3]      dobostorta.előkészítésiIdő = 25;
[4]      dobostorta.pihentetésiIdő = 20;
[5]      dobostorta.sütésiIdő = 25;
[6]      dobostorta.krémKeverésiIdő = 35;
[7]      dobostorta.összeállításiIdő = 35;
[8]
[9]      Console.Write(
         dobostorta.szummaIdőSzámít());
[10]
[11] dobostorta.szummaElkészítésiIdő= 10;
```

Az első sorban létrehozunk tehát egy példányt a Torta osztályból. Ezt a példányt *(objektumot)* dobostortának hívják. Beállítjuk az egyes részfázisokhoz tartozó időtartamokat, majd kiíratjuk a szummaldőSzámít() metódus eredményét. A fenti adatokkal ez 25+20+25+35+35 = 140 perc.

A 11. sorban azonban valami miatt a kezünkbe vesszük az irányítást, és az elkészítési időt 10 percre állítjuk. A következő használatkor *(lekérdezéskor)* valószínűleg jó alaposan meglepődünk majd, és fellelkesülünk, hiszen az itt szereplő adatok szerint egy dobostorta elkészítésére elegendő 10 perc úgy, hogy mindössze egyetlen ember van a konyhában – úgyhogy ezzel a felkiáltással sietve rávesszük lelkes Párunkat, hogy süssön nekünk tortát, mi meg közben ellazulva nézhetjük kedvenc meccsünket a TV-ben. Az eredmény valószínűleg nem dobostorta, hanem kétségbeesett siránkozás lenne, hogy ez már megint nem működik, hiszen 10 perc alatt még az előkészületek sincsenek készen a konyhában.

A problémát orvosolandó, tegyük rejtetté a szummaElkészítésiIdő attribútumot, hogy kívülről ne legyen közvetlenül hozzáférhető. Ha ezután a dobostorta attribútumait nézzük, azt láthatjuk, hogy ez eltűnt a kívülről látható elemek listájából. Mindössze a szummaldőSzámítás() metódussal férhetünk hozzá az éppen aktuális értékhez.

Milyen előnye van még ennek a metódusnak? Miért lehet hasznos a használata ahelyett, hogy minden egyes meghívását kiváltanánk az egyes időszeletek hosszának összeadásával? Valahogy így:

```
[1] Torta dobostorta = new Torta();
[2]
[3]     dobostorta.előkészítésiIdő = 25;
[4]     dobostorta.pihentetésiIdő = 20;
[5]     dobostorta.sütésiIdő = 25;
[6]     dobostorta.krémKeverésiIdő = 35;
[7]     dobostorta.összeállításiIdő = 35;
[8]
[9]     Console.Write(
        ↳  dobostorta.előkészítésiIdő +
        ↳  dobostorta.pihentetésiIdő +
        ↳  dobostorta.sütésiIdő +
        ↳  dobostorta.krémKeverésiIdő +
        ↳  dobostorta.összeállításiIdő);
```

A 9. sor itt ugyanazt az eredményt adja, mint a szummaldőSzámítás függvény, miért kell akkor még ezzel is növelni az osztály méretét?
Gondolkodjunk ismét közösen. Képzeljük el a következő szituációt: a programot felhasználó Gizike ránk szól, hogy neki nagyon hiányzik még egy kikeverésildő attribútum is, mert ezt igazából nem tudja sem az előkészítéshez, sem a pihentetéshez, sem a sütéshez sorolni. Felvesszük hát az osztályba ezt az új attribútumot, sőt még inicializálunk is egy értéket neki *(mondjuk 0-ra)*, hogy a már korábban leprogramozott és létrehozott objektumoknál se legyen gond az értelmezése. Igen ám, de Gizike elkezdi beállítani ezeket az értékeket, mondjuk a dobostortára 30 percet. Néhány óra vagy nap múlva ismét sikítva hív bennünket, hogy valami nem jó, mert ezek az értékek nem adódnak hozzá a teljes elkészítési időhöz. Kénytelen-kelletlen nekiállunk orvosolni ezt is, és fél óra után leimádkozzuk a csillagokat is az égről, hiszen minden egyes összegszámítást át kell írnunk, pontosabban kibővíteni az új időtartammal:

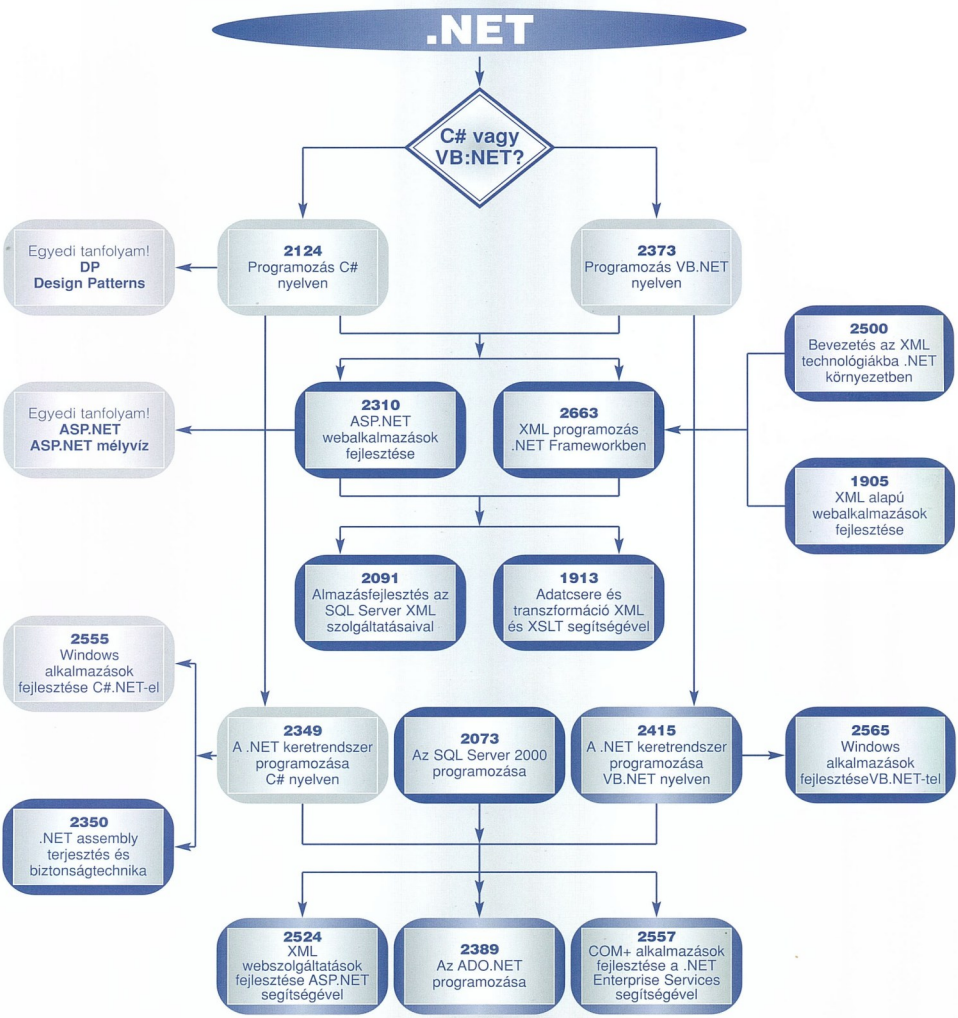
```
Console.Write(
        ↳  dobostorta.előkészítésiIdő +
        ↳  dobostorta.pihentetésiIdő +
        ↳  dobostorta.sütésiIdő +
        ↳  dobostorta.krémKeverésiIdő +
        ↳  dobostorta.összeállításiIdő +
        ↳  dobostorta.kikeverésiIdő);
```

A 9. sorhoz tehát újabb tagot adunk. És ugyanezt megtesszük a többi háromezer helyen is, ahol az adott összegre vagyunk kíváncsiak. Mennyivel egyszerűbb lett volna megvalósítani a szummaldőSzámítás függvényt, és azt meghívni ezeken a helyeken! Hiszen akkor egyes egyedül a függvény belsejét kellene most átírnunk, csak ott kellene felvennünk az újabb tagot az összegbe, és akkor mindenhol helyesen jelenne meg az érték!
Hasonló a helyzet akkor is, ha rájövünk, hogy egy tagváltozó neve nem felel meg a tartalomnak *(például időközben pontosítottuk a felhasználó igényeit)*, vagy a tagváltozóban egyéb módosításokat kívánunk bevezetni. Ha az attribútum közvetlenül elérhető kívülről is, a belső módosítás után mindene egyes hivatkozást át kell írnunk. Ha azonban a fentihez hasonlóan minden paraméterhez van külön elérőfüggvény, ez a veszély nem áll fenn, hiszen ekkor elég a függvényt módosítani, kifelé transzparens lehet a változás.

Jelenleg tehát már értjük, mik azok az osztályok, milyen adatokat kell megadnunk definiálásukkor. Semmit nem tudunk azonban még az osztályok kapcsolatáról, és egy csomó olyan gyakorlati dologról, ami szintén hasznos lehetne munkánkhoz. Ezért a következő hónapokban tovább boncolgatom ezt a témát: részletezem az osztályhierarchiák felépítésének módjait, a különböző osztályok együttműködését. Bemutatom az UML-hez tartozó többi diagramot is, és azok kapcsolatait egymással, hogy világossá váljon, hogyan épül el egy rendszer teljes UML modellje.
Addig is mindenkinek jó szórakozást kívánok az UML-lel és a hozzá kapcsolódó CASE eszközökkel történő ismerkedéshez! Egy dolgot nem szabad elfelejteni: tanulni csak úgy lehetséges, ha az ember próbálkozik!
Úgyhogy hajrá!

Molnár Ágnes
agnes.molnar@vipmail.hu

Fejlesztői tanfolyamok térképe



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.

Szeptember 15-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.

Szeptember 15-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!

MCDBA (adatbázis-adminisztrátor) képzés

a kötelező vizsgákhoz
504.000 Ft helyett már nettó **459.000 Ft-tól!**

Microsoft SQL 2000 alapú adatbázisrendszerek teljeskörű üzemeltetésével, támogatásával, karbantartásával, tervezésével és implementálásával megbízott szakemberek részére ajánlott képzés.

Szeptember 15-től!

Microsoft .NET fejlesztői képzés

774.000 Ft helyett nettó **619.000 Ft-ért!**

Fejlesztők, programozók számára ajánlott képzés, akiknek feladatuk lesz összetett, Windows-alapú és webes alkalmazások készítése, fejlesztése, szolgáltatások implementálása és tervezése. Az alapoktól a haladó szintig, párhuzamosan a C# és Visual Basic .NET programnyelveken.

Október 13-tól!

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 CERTIFIED Technical Education Center

SZÁMALK TOVÁBBKÉPZÉS

SZÁMALK

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