tech.net

working with windows

IV. / 08. szám
2003. augusztus
1395 Ft

ISSN 15865185 08

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

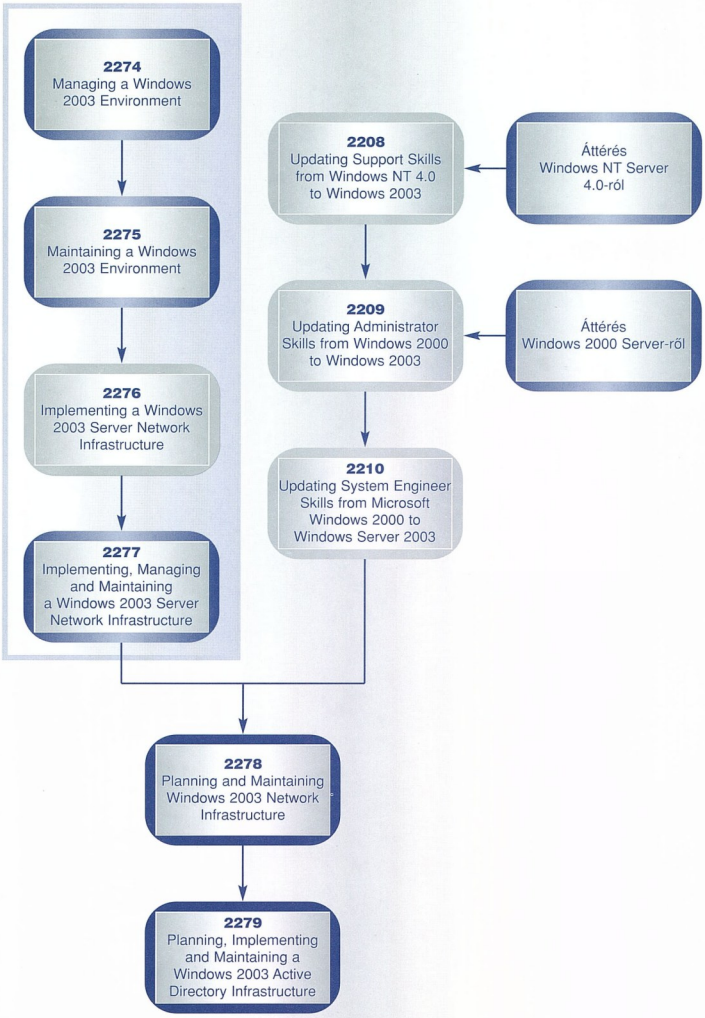
Hirdetésfelvétel: **Szívós Éva**
Telefon: 472-1214
Fax: 472-1215
info@netacademia.net

Nyomdai előkészítés:
Ars Luna Bt.
Vezető: Dobák Ildikó

Nyomda:
AduPrint Kiadó és Nyomda Kft.
1033 Budapest,
Csikós utca 8.
Felelős vezető: **Tóth Béláné**

ISSN 1586-5185

Windows 2003 Server rendszergazdai tanfolyamok térképe



Séta a koponyánk körül

Semmi sem a sajátod azon a néhány köbcentiméteren kívül, ami a fejedben van. (George Orwell)

Egy laikus számára úgy tűnhet, hogy egy informatikus szakma legfontosabb kelléke a pingvines-, vagy ablakos kávésbögre, és egy hátizsáknyi telepítő CD. Ez nem így van. Az informatikus legfontosabb munkaeszközét a nyakán hordozza, és gondos gazdaként karban kell tartania azt. Erről lesz szó rövid kis sétánk alatt. Tartsanak velem!

A cikket gondolatébresztőnek szánom. Gondolatébresztőnek és gondolkozásra serkentőnek, hiszen a téma – választott hivatásánál fogva – közvetlenül érint minden szakmabélit. Én magam talán akkor gondolkoztam rajta először, mikor egy beszélgetés során azt hallottam, hogy „jó informatikus, ő mindenhez ért". Hmm...

Hirtelen megjelent lelki szemeim előtt hősünk „IT" feliratos szupermen jelmezében, mint az informatika minden jelenleg ismert ágának feketeöves nagymestere. Ha valaki a leírás alapján magára ismer, kérem, hogy jelentkezzen *(jutalma egy csattogószárnyú papillangó).*

Hová tűnt szupermen?

Létezik a valóságban szupermenünk? Egyáltalán létezhet? Nehéz ügy, meglehetősen kicsi a valószínűsége. Hogy miért állítom ezt? Mert az informatika tudománya elérte a bonyolultságot, vagyis azt a komplexitáshatárt, hogy szaktudományokra szakadjon. *(Az egyik ilyen szaktudomány, vagyis diszciplína ismereteit feldolgozó lapot tart kezében a kedves olvasó.)*

A jelenség nem példa nélküli, gondoljunk csak a fizikára, matematikára, közgazdaságtanra, vagy az orvostudományra. A folyamatos kutatás és fejlesztés, az ismeretek és a tudásbázis szakadatlan bővülése magában hordozza az alapdiszciplína szakterületekre szakadásának szükségességét. Könnyebb ezt megérteni, ha az elejétől kezdve figyelemmel kísérjük a történetet.

Mikor egy ifjú titán ismerkedni kezd egy tudománnyal, egy nagy hegy megmászásába fog bele. A hegy maga a megismerni kívánt diszciplína, melynek alkotóelemei a kognitív sémák, a gondolkodás aktív alakuló mintái. Ezek a megismerési sémák önálló jelentéssel bírnak és önmagukban is értelmesek. Külön-külön azonban nem sok hasznukat vesszük, bonyolult és szövevényes kapcsolatrendszerük alkotja az elsajátítani kívánt szaktudományt.

Stabil alapra szükség van!

Sémák nélkül nem megy. Téglák nélkül nem épül fel a ház, ha nincsenek meg a sziklák, soha nem érünk fel a csúcsra. Mert nem lesz hová.

Bizony az alapozás fáradságos és időigényes művelet, de nem kerülhető meg. Az adott diszciplína alapjait képző sémák megismerésére szükség van, mivel a bonyolultabb sémák egyszerű sémákra épülnek. Sőt, a sémák szövevényes kapcsolatrendszere miatt nem elegendő csupán az egyes sémák bemagolása, a közöttük lévő összefüggések megismerésére is törekedni kell. Csak így „állhat össze a kép". De ne szaladjunk előre.

Az amatőrtől a Jedi lovagig

Vajon mi a különbség a nagymester és a botladozó amatőr között? Erre a kérdésre leggyakrabban azt a választ kapom, hogy „a profi többet tud". Ennyi lenne? Magoljunk és tejcsokoládé fog potyogni az égből? Vagy máshogy tudja, amit tud? Vagy más módon gondolkodik? Mitől? Nehéz rövid és frappáns válaszokat megfogalmazni ezekre a kérdésekre. Nem is lehet ez a célom. Azonban sok dolog a helyére kerülhet, ha figyelmesen tanulmányozzuk a táblázatot *(Mérő László, 2001)* a következő oldalon.

> Aki tanul, de nem gondolkodik, elveszett ember. Aki gondolkodik, de nem tanul, nagy veszélyben van.
>
> *Konfucius*

Az egyes szinteket az elsajátított kognitív sémák mennyisége határozza meg. Átlagos általános műveltséget feltételezve elmondható, hogy a legtöbb diszciplínában a kezdő, vagy haladó szintet mindenki eléri. Olyan területek sémáival is rendelkezünk, amelyek nem kapcsolódnak szorosan mindennapi munkánkhoz. Bár nem vagyunk orvosok, tudjuk mi a vérvétel, a műtét és a foghúzás. Legtöbben közülünk nem képzett közgazdászok, de ismerik a tőzsde, a részvény, a kamat fogalmát. Természetesen mivel nem vagyunk a terület szakértői, ezek a sémáink nem annyira komplexek és árnyaltak, mint amilyenekkel a diszciplína felkent nagymesterei rendelkeznek. Ez nem is meglepő, ha figyelembe vesszük a tanuláshoz, a megismeréshez és az éréshez szükséges 5-10 évet.

Akkor hogyan is volt ez a szupermen dolog? Miért nehéz mindenhez érteni? Egy nagymester néhány 10 000, az adott területhez tartozó kognitív sémával rendelkezik. Miért nem többel? A vizsgálatok szerint szakterülettől függetlenül, minden diszciplína esetében megállt ezen a szinten a sémák számának növekedése. Természetesen ezzel nem fejeződött be az egyén fejlődése, hiszen a fejében lévő sémák dinamikus, állandóan változó rendszert alkotnak. A sémái egyre árnyaltabbak és kifinomultabbak lesznek. De a számuk nagyságrendileg már nem változik. Ennek okát nem az emberi memória korlátozott kapacitásában kell keresni. A téma szakértői sze-

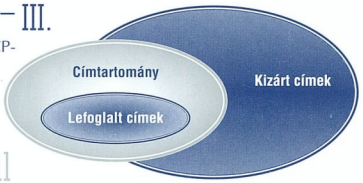
Folytatás a 4. oldalon.

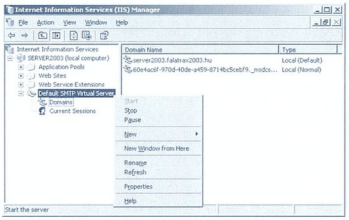
2003. 08. szám

A DHCP rejtett szépségei – III.

Előző alkalommal telepítettünk és címtartománnyal szereltünk fel egy DHCP-kiszolgálót, majd elkalandoztunk a DNS- és DHCP-integráció területére. Most még egy rövid időre visszatérünk a scope-okhoz, és alaposan szemügyre vesszük, milyen részei vannak, és milyen lehetőségekkel rendelkezünk a címtartományok kiterjesztésére.

5. oldal





Internet Information Services 6.0, III. rész

Levelező-szolgáltatások a Windows Server 2003-ban: az SMTP

Az SMTP rendszerszolgáltatás a Windows 2000-hez képest egyetlen dolog *(az LDAP útválasztás)* kivételével nem tartalmaz igazán sok újdonságot. A következő leírás így a Windows 2000 üzemeltetőinek is jó szolgálatot tehet.

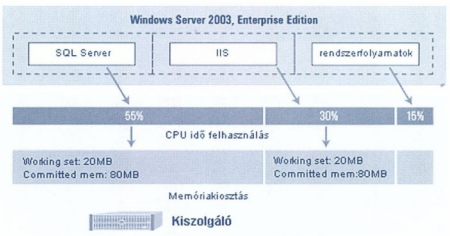
8. oldal

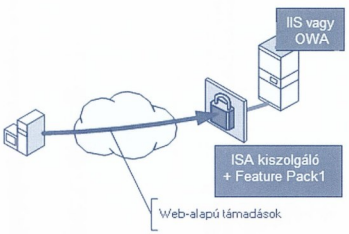
A Windows System Resource Manager

Vége az erőforrásokért folyó versengésnek!

A Windows System Resource Manager a folyamatok prioritását dinamikusan változtató algoritmus segítségével képes arra, hogy azok erőforrás-felhasználását az előre beállított határértékek alatt tartsa, így biztosítva az erőforrások optimális kihasználtságát.

14. oldal





ISA Server 2000, Feature Pack 1

Új téglák a (tűz)falban

A nemrégiben megjelent ISA Server 2000 Feature Pack 1 segítségével tűzfalunk funkcióit kiteljesednek *(kiterjedt SMTP-filter, URLScan 2.5 stb.)*, néhány mindennapos feladat megoldása sokkal egyszerűbbé válik *(például az OWA kiszolgálóik publikálása)*. És ez még nem minden!

18. oldal

Biztonságos aláírás kezelő alkalmazás készítése III.

Avagy a bürokrácia diszkrét bája

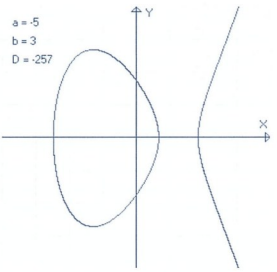
Az ember minden épeszű mértéken áthágva képes elbonyolítani tulajdon életét. Ez különösen igaz társadalmi méretekben, pláne ha az adminisztráció oldaláról közelítjük meg a jelenséget. Az elektronikus dokumentumkezelés és aláírás sajnos ezen mit sem javít: csak a hagyományos papír alapú adminisztrációt képzi le elektronikus formára. A kommunikáció felgyorsulhat és egyszerűsödhet, de a folyamatok alapvetően nem változnak. Mivel az aláírás *(akár a hagyományos, akár az elektronikus)* egy jogi szempontból is érvényes kötelezettségvállalást fejezhet ki, duplán igaz rá minden bürokratikus szabály. Nem az elektronikus aláírás bonyolult, hanem az élet, aminek próbál megfelelni.

22. oldal

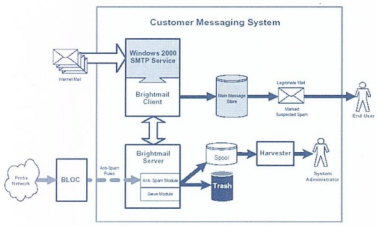
Elliptikus görbe kriptorendszerek I. rész

Elliptic Curve Cryptography – elméleti háttér

Egy jó ideje egyre több helyen találkozhatunk az ECC betűhármassal, mint egy kriptorendszer megjelölésével. Egyre több előnyéről hallunk: az RSA-nál rövidebb kulcsokkal érhető el ugyanakkora biztonság, gyorsabb a működése, kisebb a memóriaigénye. Mindezek a SmartCard technológia biztonsági eszközeit is az ECC felé fordítják. De mi is ez? Elliptic Curve Cryptosystem, bár gondolom ettől most sokan nem lettek okosabbak, nekik *(is)* szól az a kétrészes cikksorozat, melynek első része következik.



26. oldal



SPAM, anti-SPAM

avagy Sajnos Pont A Mailboxomban gyülekező nemkívánatos adathalmaz

Napjaink egyik legkellemetlenebb problémájával foglalkozom ebben a cikkben, mégpedig a nem kért, de mégis tömegével kapott e-mail üzenetekkel, amelyek elárasztanak mindenkit, aki akár csak egyszer is küldött valaha egy e-mailt bárhova a saját hálózatán kívülre.

30. oldal

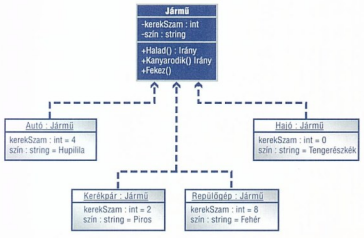
SQL Profiler

Aki SQL Serverrel foglalkozik, akár fejlesztőként, akár rendszergazdaként időről-időre találkozhat „rejtélyes" esetekkel. Amikor valami okból nem úgy fut egy alkalmazásunk, ahogy mi elvárjuk, és sehol nem találjuk a hiba okát. Ilyenkor érdemes az SQL Profilerhez fordulnunk. Ez az eszköz az SQL Server részeként sokszor segítheti munkánkat.

33. oldal

Objektum Orientált Alkalmazásfejlesztés I.

OOP, OOA, UML, DP, RUP, ER, ORM,... Sokáig folytathatnám még a HBR-eket ☺, amelyek a szoftvertervezés és -fejlesztés során felmerülnek, és amelyek tengerében nagyon könnyen el lehet veszni. Az útkeresés során látni fogjuk, hogy nem elég, ha az ember ismeri a rövidítések jelentését, a lexikális tudás mellett rengeteg egyéb képességre kell szert tennünk.



37. oldal

Folytatás a 4. oldalról.

rint a magyarázat erre egyfajta komplexitási szint elérése lehet. Amennyiben egy tudomány eléri ezt a szintet, szakterületekre szakad, melyek kinevelik saját szakértőiket és nagymestereiket. Magától értetődően lesznek közös kognitív sémák, hiszen osztanak, szoroznak a matematikusok és a fizikusok is, és a protokoll, merevlemez, kernel, DNS fogalma ismerős minden informatikával foglalkozó számára. A sémák újra felhasználhatóak, ezért igazodik el könnyebben egy szakterület ismerője egy „rokon" területen.

Mester, mit javasolsz?

A mindennapi munkavégzés során számos megoldandó problémával találkozunk. A rutinfeladatokat nem tekintem problémának, mert ezek jól ismert, gyakran jelentkező feladatok. Kidolgozott és begyakorolt forgatókönyvek alapján nem jelent szellemi teljesítményt az elvégzésük. Probléma az, amire nincs kidolgozott, kész megoldásunk.

Az egyes tudásszinteken jelentősen eltér a megoldandó problémára adott megoldási javaslatok száma. Kezdőszinten nagyon kevés a sémák száma, sokszor a probléma azonosításához sem elegendő, így nincs értékelhető megoldás. A haladó szint a legveszélyesebb, mert már ismer szakkifejezéseket, és a maga logikátlan, kevert módján minden, a témával kapcsolatos ismeretét megoldásnak véli. Rengeteg megoldást javasol, ezek túlnyomó része használhatatlan. A szakértő már rendelkezik azzal a tudással, ami könyvekből megtanulható. Felismeri a kapcsolatokat, és szakmai logika szerint racionálisan keresi a lehetséges megoldásokat. A haladónál lényegesen kevesebb, de még mindig „sok" megoldást javasol, mely megoldások szakmai ismeretekkel alátámasztottak és többnyire megfelelő választ adnak a vizsgált problémára. A „nagymester" gondolkodása mélyen intuitív, probléma megoldási módja képi, szintetikus, gyakran transzlogikus. Csupán néhány megoldást javasol, azonban ezek között gyakran található forradalmian új, más megközelítésű, „nagy ötlet", ami még soha, senkinek nem jutott eszébe, mégis tökéletes megoldása a felvetett problémának.

	Kezdő	Haladó	Szakértő	Nagymester
Kognitív sémák mennyisége	néhány 10	néhány 100	néhány 1000	néhány 10 000
Kognitív sémák minősége	bonyolult, hétköznapi inadekvát	egyszerű, adekvát, nem kielégítő	bonyolult, adekvát, szakszerű	komplex, analógiák
Problémamegoldás módja	logikus, a hétköznapi logika szerint	logikátlan, mert kevert	logikus, analitikus, a szakmai logika szerint	képi, szintetikus, gyakran transzlogikus
Szakmai kommunikáció minősége	szakszerűtlen, hétköznapi intuícióra alapoz	görcsös, hullámzó színvonalú	szakmailag korrekt, formális, tárgyszerű	mélyen intuitív, informális, áttekintő
Szakmai nyelve	nincs	nehézkes, „ideges"	szabályszerű, kifejező	„anyanyelvi", képszerű
Gondolkodási stílus	intuitív	kevert, ezért gyakran logikátlan	racionális	intuitív
Tudatosság szintje	még nem tudja, mit nem tud	tudja, mit nem tud még	tudja, mit honnan tud	tudja, mi a helyénvaló, de nem tudja honnan
Érés ideje	-	néhány év	kb. 5 év	minimum 10 év
Mi kell hozzá	érdeklődés, tanulás	folyamatos tanulás	képzettség, iskolai végzettség	tehetség

Egyetlen dolog állandó: a változás

Bár az informatika ember alkotta világ – így elméletileg teljesen megismerhető – szupermenünk dolgát tovább nehezíti a fejlődés üteme. Nemrég egy Microsoft prezentációban láttam a következő adatokat:

1993	Windows NT 3.1
1994	Windows NT 3.5
1995	Windows NT 3.51
1996	Windows NT4
2000	Windows 2000
2003	Windows 2003

Röpke 10 év, 6 szervertermék, 4 generáció. És a táblázatban csak a szervertermékek szerepelnek, nem szóltam a kliensmékekről, alkalmazásokról, fejlesztőeszközökről. Évente biztosan megjelenik egy új, meghatározó termék újdonságok *(új sémák)* tömegét zúdítva a szakmára. Mire gondolok? Íme: AD/AM, VDS, VSS, CMAK, PPPoE, PEAP, RSoP, SUS, EMS. Mindegyik ismerős? Egyiket sem kell magyarázni? És hol voltak ezek 1-2 éve?

Bizony ez nem az a „karosszékben hátradőlős" szakma. Mit jelent ez a szakemberek számára? Folyamatos tanulást.

Hol a határ?

Igen, folyamatosan kell tanulni. Figyelem! A folyamatos tanulás nem csupán ahhoz szükséges, hogy felfelé másszunk a képzeletbeli hegyen, hanem ahhoz is, hogy ne csússzunk lefelé!

Rendben, tanulunk, fejlődünk, de meddig? Jön egy fal? Bemondják, hogy „végállomás, kérjük, szálljanak le"? Vannak az emberi megismerésnek és a technikai fejlődésnek határai? Nem fogok jövendölésekbe bocsátkozni, mert ebbe nagy emberek is belesültek *(640 kByte memória mindenre elegendő!)*. Felesleges a célozgatás, mert ahhoz, hogy meg tudjuk jelölni a határokat, el kell látnunk odáig. Ezért a kérdésre inkább egy idézettel válaszolok:

Richard Bach művében, Jonathan a renitens, saját korlátait feszegető sirály távozásakor a következő tanácsot adja Fletcher sirálynak: „Ne higgy a szemednek. Amit azzal látsz, csupa korlát." Nincs határ.

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

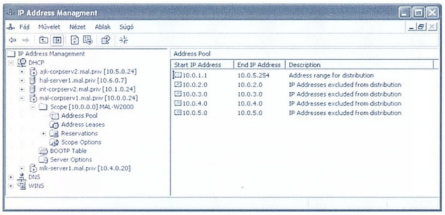
Felhasznált szakirodalom

[1] Mérő László: Új észjárások *(Tercium kiadó, 2001)*
[2] Richard Bach: Jonathan, a Sirály

A DHCP rejtett szépségei – III.

Előző alkalommal telepítettünk és címtartománnyal szereltünk fel egy DHCP-kiszolgálót, majd elkalandoztunk a DNS- és DHCP-integráció területére. Most még egy rövid időre visszatérünk a scope-okhoz, és alaposan szemügyre vesszük, milyen részei vannak, és milyen lehetőségekkel rendelkezünk a címtartományok kiterjesztésére.

A scope központi fogalom a DHCP-szolgáltatás kialakításakor. Miután a varázsló segítségével létrehoztunk egyet, érdemes alaposan megvizsgálni a kezelőfelületet, hogy lássuk, hogyan finomíthatjuk még a beállításokat.



DHCP-kiszolgáló egy scope létrehozása után

A címtartomány belülről

A scope tulajdonságairól és a legfontosabb opciókról már értekeztünk, de a kizárt és lefoglalt címekkel *(reservations)*, valamint a speciális scope tulajdonságokkal még nem foglalkoztunk.

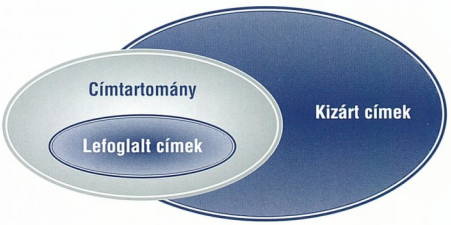
A fenti ábrán látható, hogy a címtartomány definiálása mellett azonnal meg kell adni azokat a címeket vagy cím-intervallumokat, amelyeket a teljes címtartományon belül szeretnénk kizárni a kiadható IP-címek közül. A kizárás persze nem kötelező, de néha kifejezetten hasznos. Ha például vannak állandó, statikus IP-címet igénylő kiszolgálók vagy egyéb aktív eszközök, switchek, nyomtató-szerverek, és azok címei beékelődnek egy érvényes címtartományba, a kizárás módszerével megakadályozhatjuk, hogy a DHCP-szerver egy másik ügyfél rendelkezésére bocsássa az ominózus azonosítót. Jobb helyeken, ahol ezek az eszközök eleve egy elkülönült címtartományból részesülnek, vagy lefoglalt címeket kapnak a hostok, erre külön nincs szükség. A kizárás funkció ott is használható, amikor két DHCP-szolgáltatással magasabb rendelkezésre állást szeretnénk biztosítani, – de erről majd később.

A kiadott címek listája

Nem elhanyagolható haszonnal jár, ha szeretnénk bepillantani az épp kiadott címek listájába. Ha egy állomás MAC-címét szeretnénk megtudni, vagy csak egyáltalán az IP-címére vagyunk kíváncsiak, a kiadott címek listájának táblázatát kell átböngésznünk. A konzol információkat is szolgáltat a bérletek kiadásáról és lejárati idejéről is. Ha szükséges, törölni is lehet egy rekordot, ám ez csak nagyon körültekintő módon ajánlott. Egy cím törlése hasonló hatással van a hálózatra, mint amikor egy bérlet véglegesen lejár, azzal a különbséggel, hogy a bérletet használó ügyfél aktív maradhat, és használhatja a címet, – hisz semmiféle értesítést a rekordtörlésről nem kap. A cím ráadásul felszabadul, amit éles környezetben egy másik állomás megkaphat, – ezáltal két állomás is ugyanazzal a címmel rendelkezhet. Csak akkor javaslom tehát bérlet törlését, ha az IP-címet szeretnénk felvenni a kizárt vagy a lefoglalt címek közé. Az utóbbi esetben az ügyfél oldalán mindenképp el kell végezni egy „ipconfig /renew" műveletet is.

A lefoglalt címek listája

Gyakran összekeverik a kizárt és a lefoglalt címeket, holott ez utóbbiak más célt szolgálnak. Ha szeretnénk minél több ügyfelet a DHCP-szolgáltatás kliensei között tudni, de néhány esetben bizonyosan ugyanazt az IP-címet kellene kiadnunk egy-két állomásnak, a lefoglalt címek közé kell felvennünk az ügyfél adatait, és mindkét kívánságunk teljesült. A lefoglalt címek funkció azért működik, mert a DHCP képes a címigénylő csomagból megállapítani az adott állomás hardver-címét *(Ethernet hálózat esetén ez a MAC address)*, és ha talál ehhez a bejegyzéshez egy lefoglalt címet, azt fogja minden esetben kiajánlani. A reservations működési elvéből következik, hogy ha egy címet kizártunk, akkor az nem lehet egyben lefoglalt cím, és fordítva, a lefoglalt címeket nem szabad kizárni. Bár látszólag triviális dologról van szó, mégis nagyon gyakori konfigurációs hiba, hogy a lefoglalt címeket megpróbálják kizárni, „nehogy véletlenül is más kapja meg", vagy mert „a lefoglalt cím nem játszik, az másra nem használható, tehát ki kell zárni". Mindez a fogalmak pontatlan ismeretéből fakad. A három címtípus egymáshoz való viszonyát egy ábrán is megjelenítettem, hogy az olvasó már soha többé ne tartozzon az ilyen hibákat elkövetők közé.



A címtartomány, a lefoglalt címek és a kizárt címek viszonya

Visszatérve a lefoglalt címekhez: ismernünk kell, miként viselkedik a kiszolgáló, ha olyan címek közül foglalunk le egyet, amely korábban a címtartomány normál címei közé tartozott, és amelyet esetleg egy másik állomás már igénybe vett. Ilyenkor vagy meg kell várnunk a bérlet végét, vagy az adott ügy-

felet rá kell szorítanunk, hogy „engedje el" a címet. NT/2000/XP rendszereknél az „ipconfig /release" paranccsal érhetjük el ezt, a Win9x világban pedig a winipcfg névre hallgató kis grafikus program siet a segítségünkre. Egyébként a bérlet kiadása szintén nem automatikus, – vagyis attól, hogy a lefoglalt címet rögzítettük, még nem kapta meg azt az ügyfél. A kliensgépen kiadott „ipconfig /renew" parancs azonban általában meghozza gyümölcsét: a friss lefoglalt címet.

A lefoglalt cím nagyon hasznos szolgáltatás. Egy hálózatban rengeteg olyan állomás működhet, amely állandó IP-címet igényel. Háromszáz munkaállomáshoz már akár féltucat switch, 10-20 hálózati nyomtató, több szerver tartozhat, amelyeket mind-mind állandó címmel kell ellátni. Egy IP-cím vagy hálózati maszk-váltás rengeteg munkával jár, ha statikus címekkel dolgozunk. A manuális munka ugyanakkor mindig magában rejti a hibázás lehetőségét is. Könnyen kimaradhat egy DNS-cím vagy az idő-szerverek megadása, s máris megjósolhatatlan hibák tűnnek fel a rendszerben. A lefoglalt címekkel mindez elkerülhető, a változások bizonyosan érvényre jutnak. Hasonló módon lehet leszerelni azokat a megoldásszállítókat, akik egy monitorozó, tarifikáló vagy egyéb berendezést szállítanak, s kötik az ebet a karóhoz a statikus IP-címért. Bőven megteszi egy lefoglalt cím is. Egy tökéletes hálózatban szinte csak a WINS és DHCP-kiszolgálók rendelkeznek statikus címekkel.

Műveletek a címtartománnyal

Ahhoz, hogy egy scope ellássa feladatát, aktiválni kell. Csak aktív scope bocsát ki IP-bérleteket. A művelet rendkívül egyszerű, csupán a jobboldali egérgombbal kell kattintani a címtartományon, majd az aktiválást választani. A scope deaktiválásával megszűnik a korábbi funkciója. Csak nem aktív címtartományt lehet törölni.

A címtartomány körül a legnagyobb felhajtás akkor keletkezik, amikor a rendelkezésre álló címek elfogynak. A bérletidő rövidítésével lehet még orvosolni a címéhséget, a hosszú távú megoldás azonban mindenképpen a scope átalakítása.

Elsőre azt gondolhatnánk, hogy kényelmesen átállítjuk a címtartomány tulajdonságlapján a kezdő és a végső címet, sőt akár az alhálózati maszkot, és már készen is vagyunk. Sajnos, ez egy járhatatlan út. A szabvány szerint nem nagyon lehetne nyomon követni, hogy melyik bérletet milyen címtartomány-paraméterekkel adtuk ki. Sok kiadott cím kikerülhetne az érvényes bérletintervallumból, esetleg érvénytelen lenne az alhálózati maszk, – egyszóval kavarodás lenne. Három lehetőségünk maradt a probléma megoldására:

- a scope kiterjesztése
- az alhálózat újrakonstruálása *(resubnetting)*
- Superscope-ok alkalmazása *(multinetting)*

A címtartomány kiterjesztése

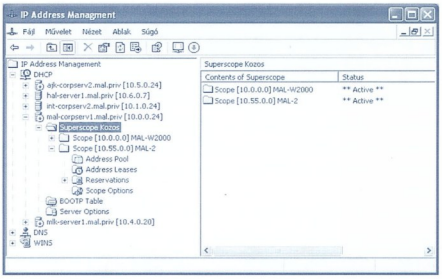
A scope megváltoztatásának egyetlen járható útja a kiterjesztés. Ha az eredeti címtartomány kisebb, mint az alhálózati maszkja alapján az lehetséges lenne, akkor mód van a tartomány végső címének kitolására. Például a 192.168.0.1 - 192.168.0.100 címtartományunkat 255.255.255.0 maszkkal, kiterjeszthetjük 192.168.0.254-ig mindenféle káros következmény nélkül. A kiterjesztés lehet többlépcsős is, és az alhálózat elméleti határáig tarthat. Korábban *(a Windows NT 4.0 szervernél)* csupán 32-esével lehetett a tartományt kiterjeszteni, vagyis 100-ról 132-re kellett volna növelni a tartomány határát. A SP6-tól kezdve ez a kötöttség már nem létezik. Amennyiben eleve belefoglaltuk a scope-ba az alhálózat engedte összes címet, ezt a módszert nem alkalmazhatjuk.

Az alhálózat újrakonstruálása *(resubnetting)*

Ha egy subnet maskot úgy alakítunk át, hogy az egyre kisebb számokat tartalmaz, akkor az alhálózatban rendelkezésre álló címek száma megnő. Nagyszerű, épp ezt szeretnénk. Csakhogy ennek komoly ára van. Az alhálózati maszk meghatározza az alhálózatot, vagyis: új maszk, új hálózat. Ezt az új hálózatot meg kell ismertetni valamennyi útválasztónkkal, át kell állítanunk az összes statikus IP-címmel rendelkező állomásunkat, továbbá el kell dobnunk a már létező scope-ot és az összes bérletet, létre kell hozni egy újat. Erre azért van szükség, mert egy scope az alhálózati maszkja alapján csak egyetlen logikai alhálózatnak szolgáltat címet, kettőnek nem. Ha sok géppel dogozunk, és nem kivitelezhető a művelet rövid idő alatt, akkor biztosítani kell a két hálózat közötti kommunikációt az alapértelmezett átjárók átkonfigurálásával. Summa summarum, az alhálózati maszk megváltoztatása hatékony ugyan, de nem sétagalopp.

Superscope-ok alkalmazása

A superscope a Windows NT4 SP2 verziója óta ismert fogalom, lényegében a címtartományok egybefogását jelenti.



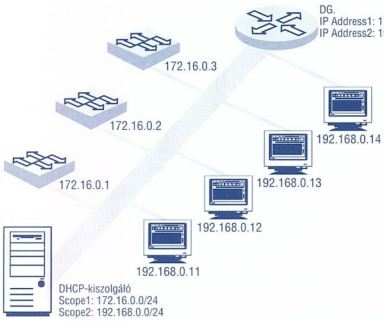
Két címtartomány alkotta superscope

Ha szeretnénk egyetlen fizikai alhálózaton több logikai *(IP)* alhálózatot üzemeltetni, azt csupán scope-ok segítségével nem könnyű megvalósítani. A DHCP-kiszolgáló minden ügyfélnek a legelőször aktivált címtartományból hajlandó címet adni. Csupán azt tehetnénk, hogy egy második hálózati kártyát is üzembe helyeznénk, majd ellátnánk az új alhálózatból származó címmel, végül létrehoznánk a második scope-ot is. Minden további *(logikai)* alhálózat egy további hálózati kártyát igényelne. Ettől a barkácsolástól kímél meg minket a superscope.

Tegyük fel, hogy van egy hálózatunk, ahol minden aktív eszköz egyben DHCP-ügyfél is. Szeretnénk elkülöníteni az aktív elemeket cím szerint is a munkaállomásoktól. Mivel a switchekkel csak a hálózati adminisztrátorok kommunikálnak az IP-címeken keresztül, a nem túlságosan nagy forgalom miatt az elkülönítés után nem keletkezik szűk keresztmetszet. A superscope segítségével a feladat megoldása könnyű.

Először létre kell hoznunk egy 192.168.0.11-192.168.0.254 címtartományt a megfelelő alhálózati maszkkal és egyéb opciókkal. Ezután el kell készíteni egy második scope-ot is, amely a 172.16.0.0/24 hálózatot foglalja magában, ám rögtön

valamennyi címet ki kell zárni, az első hármat kivéve. Ez utóbbiakat mint előre lefoglalt címet kell rögzíteni a hálózati kapcsolók MAC címével együtt. Végezetül létre kell hozni egy superscope-ot, majd a két korábbi címtartományt a superscope tagjává kell tenni.



Egy alhálózat két logikai IP-hálózattal

Ezzel készen is vagyunk. A két alhálózat közötti kommunikációt az alapértelmezett átjárón felvett útvonalbejegyzés segítségével lehet biztosítani, szükség esetén szűrve az útvonalat használó állomások számát, címét. *(Csak zárójelben jegyzem meg, hogy a superscope-ok nem csak a helyi hálózatokon támogatják a több logikai alhálózatot, hanem távoli – routereken túli – hálózatokon is. Ezeket a képességeket a DHCP Relay Agent tárgyalásakor még érintjük. A fenti 172.16.0.0/24 jelölés a továbbiakban többször szerepel majd. Azt jelenti, hogy az alhálózati maszkban 24-db 1-es szerepel, ami decimálisan 255.255.255.0-t jelent.)*

A superscope-ok fenti képességei hasznosak a címtartományok kiterjesztésekor. Elegendő egy második címtartományt definiálni, majd az első scope-pal egy superscope-ot alkotni, a címtartomány kiterjesztése máris megtörtént.

A címtartományok összefogása az alhálózat újrakonstruálásakor is jó szolgálatot tesz. Az új alhálózat bevezetése egy superscope segítségével a legegyszerűbb. Bele kell foglalni az új és a régi címintervallumokat, a régi scope-ot pedig úgy kell módosítani, hogy valamennyi címét ki kell zárni. Az ügyfelek újraindulásakor mindegyik az új intervallumból kap majd címeket. A statikus állomások, valamint az útválasztók konfigurálását nem lehet megúszni.

Superscoping, multinetting, supernetting

MCP vizsgára készülő szakértő-jelöltek tapasztalhatták, hogy néhány vizsgakérdés előszeretettel próbál zavart kelteni a fenti fogalmakkal kapcsolatban a vizsgázók fejében. Az alhálózatok kialakítása és a superscope fogalmának ismeretében most megragadom az alkalmat, és amennyire lehetséges, tisztázom az egymással részben összefüggő kifejezéseket.

A multinetting azt a szituációt jelenti, amikor egy fizikai alhálózaton több logikai *(IP)* hálózatot hozunk létre. A multinetting elméletileg nem kötődik a DHCP fogalmakhoz, így a superscope-hoz sem, elképzelhető ugyanis multinet DHCP nélkül. Gyakorlatilag azonban ilyen szituáció csak kivételes esetekben fordulhat elő, így a multinetting szinte mindig a superscope fogalmával együtt jelenik meg, hisz a superscope a multinet DHCP-támogatását takarja.

A superscoping és a supernetting összehasonlításának az az alapja, hogy lényegében ugyanazt a célt szolgálja mindkét eljárás, habár teljesen eltérő módon közelítik meg a problémát.

A superscoping alapvető célja a DHCP-ügyfelek által felhasználható címek számának növelése az alhálózati maszk változtatása nélkül. Ennek érdekében újabb IP-alhálózatokat alkalmaz, és egy közös superscope-ot hoz létre. A megoldás előnye, hogy viszonylag gyors és egyszerű módszer, a meglévő IP-címek nem változnak, és igénytelen megoldás, amennyiben nem szükségszerű, hogy a címtartományok egybefüggőek legyenek. A 192.168.0.0/24-hez hozzá lehet fűzni a 192.168.3.0/24-et, de akár a 172.16.0.0/24-et is. A módszer hátránya, hogy szűk keresztmetszetet hoz létre, hiszen a különböző alhálózatok között az adatforgalom az alapértelmezett átjáron keresztül áramlik, továbbá az állomásoknak nincs közös broadcast címe.

A superscoping és a supernetting összehasonlításának az az alapja, hogy lényegében ugyanazt a célt szolgálja mindkét eljárás, habár teljesen eltérő módon közelítik meg a problémát.

A supernetting célja ugyancsak a kiosztható címek növelése, de a fenti megoldástól eltérően az alhálózati maszk változtatásával. Minél kisebb számok szerepelnek a maszkban, annál több host-cím áll rendelkezésre. A 255.255.255.0 maszk 254 lehetséges állomáscímet jelent, míg a 255.255.252.0 már több mint négyszer ennyit. Ha az eredeti címtartomány 192.168.0.0/24 volt, az új pedig 192.168.0.0/22, akkor a 192.168.0.1-192.168.0.254 címintervallum 192.168.3.254-ig bővül.

A supernetting egy folytonos címtartományt eredményez, a DHCP támogatása pedig megoldható egyetlen scope-pal. Nem keletkezik szűk keresztmetszet, és van közös broadcast cím is. Hátránya a módszernek, hogy minden cím megváltozik, még akkor is, ha egy host névlegesen ugyanazt a címet kapja vissza. A 192.168.0.100/24 más jelent, mint a 192.168.0.100/22.

Összegzésül azt mondhatjuk, hogy a superscoping és a supernetting ugyanazt a célt *(felhasználható címek növelése)* eltérő módszerekkel, előnyökkel és hátrányokkal valósítja meg. A konkrét szituáció és preferenciák határozzák meg, hogy melyik módszer a célravezetőbb.

Lepenye Tamás
MCSE 2000
lepenyet@mal.hu

Fontosabb források és ajánlott irodalom:

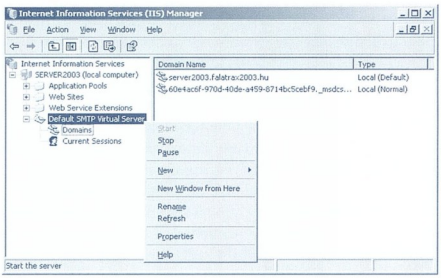
Q161571 Using DHCP "Superscopes" to Serve Multiple Logical Subnets
Q163055 DHCP Client May Fail with WinNT 4.0 SP2 Multinetted DHCP Server
Q169140 Using DHCP to Assign IP Addresses to Secondary Networks
Q174051 DHCP Server Fails to Lease Addresses for New Scope
Q255999 Increasing the Number of IP Addresses on a Subnet

Internet Information Services 6.0, III. rész

Levelező-szolgáltatások a Windows Server 2003-ban: az SMTP

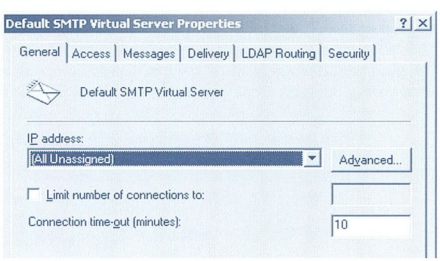
Az SMTP szolgáltatás

Az SMTP rendszerszolgáltatás a Windows 2000-hez képest egyetlen dolog *(az LDAP útválasztás)* kivételével nem tartalmaz igazán sok újdonságot. A következő leírás így a Windows 2000 üzemeltetőinek is jó szolgálatot tehet.



Az SMTP MMC konzolja – itt kezelhetjük a virtuális kiszolgálókat és az általuk kiszolgált tartományokat

Egy számítógépen természetesen egynél több virtuális kiszolgáló is futhat párhuzamosan, ha egymástól különböző IP-címen és/vagy portcímen érjük el őket. A portcím megváltoztatása azért nem szerencsés, mert a levelezőrendszerek világszerte a szabványos SMTP portot, a TCP 25-öt használják.



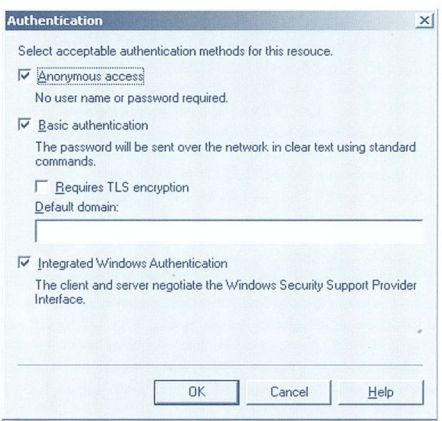
A virtuális kiszolgáló általános tulajdonságai

A virtuális kiszolgáló tulajdonságlapjának „General" oldalán határozhatjuk meg a kiszolgáló IP címét *(illetve az Advanced gombra kattintva a használt portcímet is)*. A „Limit number of connections to:" mező beállításával az SMTP kiszolgáló felé nyitható egyidejű kapcsolatok számát határozhatjuk meg. Ha a kapcsolatok száma elérte a maximumot, a kiszolgáló a további beérkező kérelmeket elutasítja. A Connection time-out: mezőben korlátozhatjuk az SMTP kapcsolatok maximális időtartamát *(egy kapcsolaton keresztül több levél is elküldhető, amíg a kapcsolat fennáll)*.

Ugyanitt találjuk, bár az ábrán már nem látszik a naplózás beállításait. Az „Enable logging" mezőben bekapcsolhatjuk a naplófájlok készítését, a Properties gombra kattintva

Biztonsági beállítások

A tulajdonságlap következő oldalán *(„Access")* a kiszolgálóval történő kommunikáció biztonsági beállításait találjuk. Az első gomb *(„Authentication")* a bejelentkezési protokollok beállításaihoz vezet.



Bejelentkezési protokollok. Az alapértelmezés csak az „Anonymous"

A Windows Server 2003 telepítésekor ezen a panelen csak a névtelen hozzáférés *(Anonymous access)* van engedélyezve, azaz a kiszolgálóra biztonsági okokból nem lehet bejelentkezni. A *(saját domainbe)* beérkező leveleket a kiszolgáló így is elfogadja. Mivel azonban a levéltovábbítási *(mail relay)* beállítások az idegen tartományba küldött levelek küldését előzetes bejelentkezéshez kötik *(lásd kicsit később)*, a „normális" működéshez itt be kell állítanunk a lehetséges bejelentkezési protokollokat is:

- **Basic Authentication:** a protokollok szabványos felhasználóazonosítási módja, ahol a felhasználónév és a jelszó a hálózaton titkosítatlanul utazik. Ez nyilvánvalóan nem biztonságos, ezért ajánlott bekapcsolni a TLS

titkosítás kikényszerítését *(„Requires TLS encryption”)*. A titkosított TLS csatorna kiépítéséhez azonban egyrészt kiszolgálótanúsítvány, másrészt további beállítások szükségesek *(erről a következő pontban lesz szó)*. Ha a bejelentkezéskor a felhasználó tartományt nem, csak felhasználónevet és jelszót ad meg, a Windows a saját tartományában fogja keresni a felhasználót. Ha azt szeretnénk, hogy az alapértelmezett keresési tartomány ettől különböző legyen, megadhatjuk a „Default domain” mezőben.

- Integrated Windows Authentication: Windows környezetben *(Windows SMTP kiszolgálók illetve ügyfél között)* jobb megoldás a Windows beépített azonosítási protokolljának *(Windows Security Support Provider Interface – SSPI)* használata.

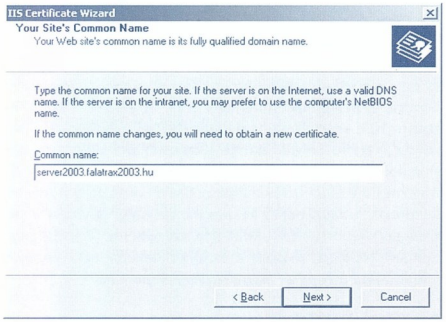
Ha az Integrated Windows és a Basic Authentication egyaránt engedélyezve van, a Windows az Integrated azonosítást fogja előnyben részesíteni.

A TLS kommunikáció

A TLS az SSL-hez hasonló *(sőt, tulajdonképpen annak utódjának tekinthető)* titkosított csatorna, ahol a csatorna kiépítéséhez szükség van a kiszolgáló tanúsítványára. *(Ez a tanúsítvány a titkosító kulcs továbbításán kívül még a kiszolgáló hiteles azonosítására is alkalmas.)* Az SMTP kiszolgálóval való kommunikáció titkosítására is a TLS-t használjuk, de a fentiek értelmében mielőtt ilyen csatorna épülhetne, a kiszolgálónknak rendelkeznie kell egy megfelelő nyílt kulcsú tanúsítvánnyal.

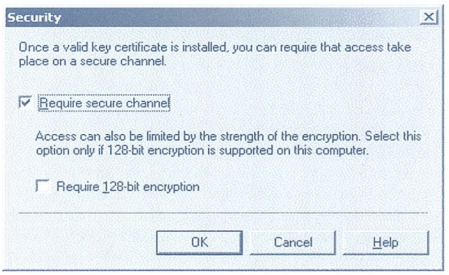
A virtuális kiszolgáló tulajdonságlapjának „Access” oldalán, a „Secure communication” mezőben két gombot találunk. A „Certificate” nyomógomb a kiszolgálótanúsítvány kezelésére használható, a „Communication” gomb segítségével pedig a majdan létrehozott TLS csatorna beállításait érhetjük el. Ez utóbbi nyomógomb egészen addig tiltott, míg a kiszolgáló nem rendelkezik érvényes tanúsítvánnyal.

Ha a „Certificate” nyomógombra kattintunk, elindul a „Web Server Certificate Wizard”, ami segít nekünk a tanúsítvány kérésében *(egyébként később a kezelésében, sőt törlésében is)*. Ha új tanúsítványt kérünk *(ehhez természetesen szükség lesz majd egy tanúsítványkiadó [CA] szolgáltatásra is, de az egy másik cikksorozat témája)*, arra vigyázzunk, hogy a tanúsítvány a kiszolgáló internetes *(DNS)* nevére szóljon, különben a tanúsítvány ellenőrzésekor azt az ügyfelek nem fogadják majd el.



A SMTP-kiszolgáló tanúsítványában a kiszolgáló teljes és érvényes DNS-neve szerepeljen

Ha a kiszolgáló tanúsítványa elkészült, a Communications gombra kattintva a következő dialógusablakot látjuk:



A TLS beállítások dialógusablaka

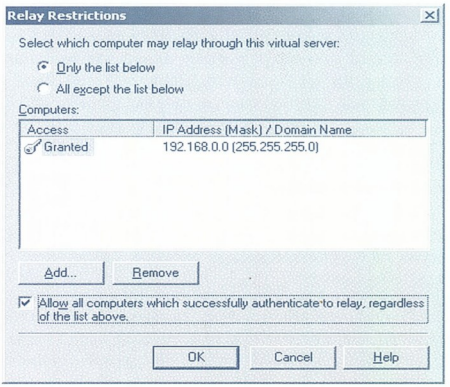
A tanúsítvány telepítésétől kezdve a TLS csatorna kiépíthető, de használata nem kötelező. Ha azonban itt *(ahogy az ábrán is látható)* bekapcsoljuk a „Require secure channel” opciót, titkosítatlan kapcsolat egyáltalán nem jöhet létre a kiszolgálóval. Egy Internetes üzemben is működő SMTP szolgáltatásnál éppen ezért ezt ne kényszerítsük ki *(itt)*, hiszen az Internetről érkező levelek kiszolgálói ezt nem biztos, hogy támogatják.

Ha az azonosításhoz szeretnénk a titkosított csatornát, jobb, ha az előző *(„Authentication”)* dialógusablakban, a Basic Authentication mezőben kapcsoljuk be a TLS-t, ekkor ugyanis csak akkor igényel a kiszolgáló titkosított kapcsolatot, ha erre az azonosításra sor kerül.

Kapcsolódás, Mail Relay

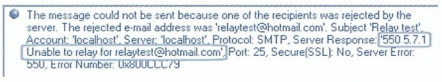
A kiszolgáló tulajdonságlapjának „Access” oldalán található következő nyomógomb a „Connections” feliratot viseli. A gombra kattintva beállíthatjuk, hogy a kiszolgálót milyen IP címekről lehet elérni *(az alapértelmezés az, hogy mindenhonnan)*. Ha tudjuk, hogy az SMTP szolgáltatást csak adott IP-cím tartomány*(ok)*ból érik el, itt beállíthatjuk ezt a korlátozást. Ha azonban a kiszolgálónk az Internetről is fogad leveleket, hagyjuk az alapértelmezést.

A következő nyomógomb a „Relay...”.



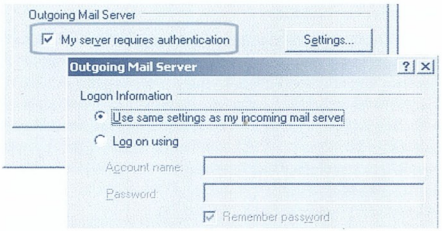
Levéltovábbítási (Mail Relay) korlátozások. Alapértelmezésben a fenti lista teljesen üres!

Az előbbi, „Relay Restrictions" ablakban a levéltovábbítást *(Mail Relay-t)* szabályozhatjuk. Az SMTP szolgáltatás elfogad és kézbesít minden bejövő levelet, amelynek címzettje valamelyik, általa kezelt tartományba tartozik. Ha azonban a címzett postaládája egy másik kiszolgálón található, a levelet az SMTP szolgáltatásnak továbbítania kellene. Ez utóbbi lépés azonban már nem általánosan engedélyezett, hiszen ezt a „trükköt" szeretik kihasználni világszerte a kéretlen reklámlevelek kiküldéséhez. Az SMTP szolgáltatás csak akkor hajlandó erre, ha a „külső" levelet küldő ügyfél a fenti dialógusban felsorolt IP-cím tartományból érkezik *(ez a lista frissen telepített kiszolgálónál üres, ide kézzel felvehetjük a belső hálózatunk IP címeit)*, illetve, ha a levél küldése előtt az ügyfél felhasználónevével és jelszavával bejelentkezik *(„Allow all computers which successfully authenticate to relay…")*. A bejelentkezés protokollja a korábban bemutatott „Authentication" dialógusban állítható be *(emlékszünk: Basic vagy Windows Integrated)*.
A bejelentkezés szükségességéről azonban értesítenünk kell a felhasználókat, különben a levél elküldésekor a kiszolgáló hibával visszautasítja őket:



A tipikus hibaüzenet: „Unable to relay…"

A bejelentkezéshez a levelezőprogramban be kell állítanunk a kimenő kiszolgálóhoz történő bejelentkezést:

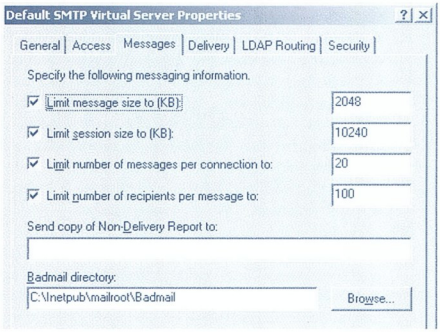


A levelezőprogramban kapcsoljuk be a kimenő kiszolgálóhoz történő bejelentkezést!

A bejelentkezés használhatja a beérkező levelek letöltéséhez megadott felhasználónevet és jelszót *(„Use same settings as my incoming server")*, de megadhatunk ettől teljesen eltérő adatokat is *(„Log on using…")*.

Az üzenetek beállításai

Térjünk vissza a virtuális kiszolgáló beállításaihoz! A kiszolgáló tulajdonságlapjának „Messages" oldalán a beérkező üzenetek és kapcsolatok korlátozásait állíthatjuk be *(az ábrán az alapértelmezések)*:



A beérkező levelek beállításai

Limit message size to: A beérkező levelek maximális mérete egyenként *(2MB)*. A kiszolgálónkhoz csatlakozó távoli SMTP kiszolgáló a levél küldése előtt ezt az értéket lekérdezheti, és ha a küldendő levél mérete meghaladja az itt beállított értéket, a levelet meg sem kísérli elküldeni *(erről egy kézbesítetlenségi jelentésben [Non Delivery Report, NDR] értesíti az eredeti feladót)*.
Ha a kiszolgáló nem kérdezi le a maximális levélméretet, és megkísérli elküldeni nekünk a levelet, az SMTP szolgáltatás az itt beállított határ elérése után hibaüzenettel megszakítja a kapcsolatot:

A túl nagy méretű leveleket a kiszolgáló visszautasítja

Ha két SMTP kiszolgáló egymással felépített egy kapcsolatot, a hálózati terhelés csökkentése érdekében egy kapcsolaton keresztül több üzenetet is elküldhetnek *(egymás után)*. **A Limit session size to:** mezőben az egy kapcsolaton keresztül beérkezett levelek összesített méretét állíthatjuk be *(ebbe csak a levelek törzsének mérete számít, a fejlécinformációk nem)*. Az alapértelmezés itt 10MB, ez tehát az egy kapcsolaton keresztül *(„egy szuszra")* beküldött levelek összesített mérete. Ugyanide tartozik a következő opció ***(Limit number of messages per connection to:)***, ahol az egy kapcsolaton belül beküldhető üzenetek maximális száma *(10)* határozható meg. Az utolsó opció ***(Limit number of recipients per message to:)*** az egy üzenetben található címzettek maximális számát korlátozza. A beállítható minimális érték itt az SMTP szabvány szerint 100, ezért ez az alapértelmezés is. A korlátok elérésekor az SMTP kiszolgáló bontja a kapcsolatot illetve visszautasítja a kérdéses levél kézbesítését.
A kézbesíthetetlen üzenetekről az SMTP kiszolgáló az eredeti feladónak hibaüzenetet, kézbesíthetetlenségi jelentést *(Non-Delivery Report, NDR)* küld. Ha szeretnénk, a szolgáltatás minden NDR-ről másolatot küldhet nekünk egy megadott címre ***(Send a copy of Non-Delivery Report to)***.

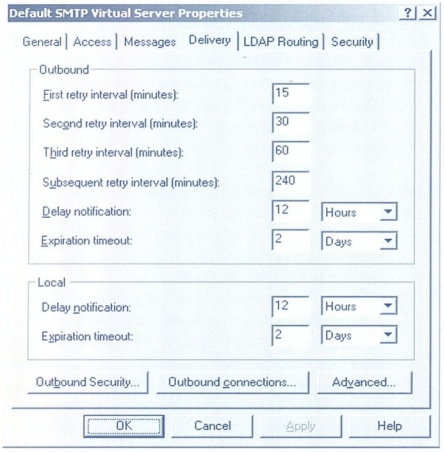
A Badmail könyvtár

Az NDR-t a kiszolgáló ugyanúgy kézbesíti, mint bármilyen már e-mailt. Ha az NDR kézbesíthetetlen, arról természetesen

újabb NDR már nem készül, hanem az üzenet az SMTP kiszolgáló „Badmail" könyvtárába kerül.
A könyvtár alapértelmezésben „C:\InetPub\mailroot\Badmail" elérési úton található, de a **Badmail directory** ablakban ezt módosíthatjuk.
Ha a kiszolgálónk *(például hibás Mail Relay beállításoknak köszönhetően)* kéretlen reklámlevelek százezreit próbálja kiküldeni *(ezek közül jópárat persze hibás címre)*, minden egyes próbálkozásról NDR készül. Mivel azonban az ilyen reklámlevelek feladói a legritkább esetben érvényes és létező címek, az NDR *(sok-sok társával együtt)* előbb vagy utóbb a Badmail könyvtárban köt ki. Ezért ezt a könyvtárat érdemes az operációs rendszertől elkülönített partícióra tenni, esetleg kvótával korlátozni, mielőtt megtölti a rendszerpartíciót, működésképtelenné téve ezzel a teljes számítógépet.

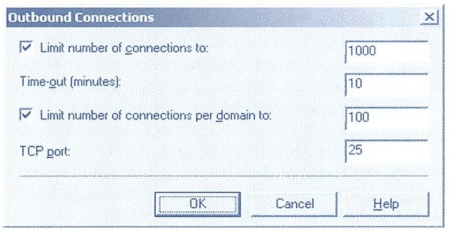
A levelek kiküldésének beállításai

Ha a kiszolgáló a kimenő levelet elsőre nem tudja továbbítani *(mert például a távoli kiszolgáló nem érhető el)*, adott ideig időközönként újra próbálkozik *(erről néha értesíti a feladót)*, egészen addig, míg fel nem adja a próbálkozást *(erről a tényről természetesen ugyancsak NDR készül)*.



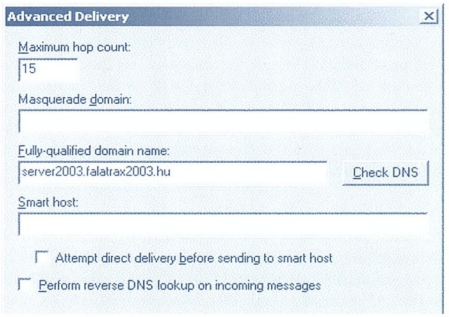
A levelek kiküldésének beállításai

A kimenő levelek újrapróbálkozási időtartamai ***(... retry interval)*** 15, 30, 60, majd 240 perc. A kézbesítési próbálkozásokról a rendszer 12 óránként értesíti a feladót ***(Delay notification)***, a próbálkozást pedig 2 nap múlva adja fel véglegesen ***(Expiration timeout)***.
Az Outbound Security... nyomógombra kattintva a kimenő levelek küldéséhez szükséges bejelentkezési információkat adhatjuk meg. A kézbesítés alapértelmezésben bejelentkezés nélkül *(Anonymous)* történik, de szükség esetén itt beállíthatjuk a bejelentkezési protokollokat *(Basic, Integrated Windows)*, illetve a használatukhoz szükséges felhasználónevet és jelszót, valamint a TLS-csatorna használatának kikényszerítését is.
Ezek a beállítások az egész virtuális kiszolgálóra érvényesek. Később céltartományonként ettől eltérő beállításokat is megadhatunk majd.



A kimenő kapcsolatok beállításai

Itt a kiszolgáló által egyidejűleg fenntartott kimenő kapcsolatok számát korlátozhatjuk ***(Limit number of connections to)***, illetve egy-egy kapcsolat maximális várakozási időtartamát ***(Time-out)***. Ugyanitt állítható be a használt **TCP port** *(25)*. Ez utóbbit általában nem ajánlott változtatni, hiszen a kiszolgálók Internetszerte ezen a porton várják a kapcsolatokat. A **Limit number of connections per domain to** mezőben azt határozhatjuk meg, hogy tartományonként *(azaz címzett kiszolgálónként)* egyidejűleg mennyi kimenő kapcsolatot építhet fel az SMTP szolgáltatás.



A kimenő kapcsolatok további beállításait az **„Advanced"** gombra kattintva látjuk:
Az internetes leveleket a kiszolgálók egymás között adják-veszik, míg az el nem éri a címzett kiszolgálóját. Azt elkerülendő, hogy a levelek a végtelenségi keringjenek a hálózatban, a **Maximum hop count** mezőben beállítható, hogy a levél mennyi ilyen kiszolgálók közötti továbbítás után számít kézbesíthetetlennek. Ilyenkor az üzenetet a kiszolgáló már nem továbbítja, a feladónak pedig NDR-t küld.
A **Masquarade domain** mezőben megadhatunk egy domainnevet. Az SMTP-kiszolgáló ekkor erre a tartománynévre cseréli a levelek kiküldésekor használt Mail From: mezőben található egyéb tartományneveket. Ez csak az SMTP-levelek forráskódjában látszik, a felhasználó által látható feladó neve és címe továbbra is az lesz, amit a levelezőprogramban meghatároztak.
A **Fully-qualified domain name** mezőben a kiszolgáló teljes DNS-nevét adjuk meg. Az SMTP-szolgáltatás ezt a címet használja a levelek továbbításakor kitöltött fejlécekben, illetve így köszönti a beérkező SMTP-kapcsolatokat is.
Alapértelmezésben a kiszolgáló a DNS-szolgáltatás segítségével minden kiküldött levél címzett tartományához megkeresi a felelős SMTP-kiszolgálót, és közvetlenül felveszi vele a kap-

csolatot. Ha azonban a kiszolgálónk egy tűzfal mögött van, beállíthatjuk úgy is, hogy minden kimenő levelet egy másik *(SMTP)* kiszolgálónak továbbítson, és majd ez a második szolgáltatás gondoskodjon az üzenet kézbesítéséről. A kiszolgáló DNS-nevét a Smart Host mezőbe írjuk be. Ha IP címet szeretnénk megadni, írjuk azt **[kapcsos zárójelek közé].** Ha bekapcsoljuk az **Attempt direct delivery before sending to smart host** lehetőséget, az SMTP-szolgáltatás minden esetben megpróbálkozik a közvetlen kézbesítéssel, és csak akkor továbbítja a levelet a **Smart Host** felé, ha a közvetlen kapcsolat nem építhető fel.

Az utolsó opció a **Perform reverse DNS lookup on incoming messages**. Ha ezt bekapcsoljuk, az SMTP-szolgáltatás minden bejövő kapcsolat megnyitásakor ellenőrzi, hogy a távoli kiszolgáló által a kapcsolatban *(a HELO parancsban)* megadott DNS-név megfelel-e annak az IP-címnek, ahonnan a kapcsolatot a valóságban felépítették. Bár ez biztonsági szempontból jónak tűnhet, vegyük figyelembe, hogy ebben az esetben minden egyes beérkező kapcsolat egy DNS-lekérdezést eredményez, ezért lassítja a kiszolgáló működését.

Az SMTP szolgáltatás hozzáférési jogosultságai

A kiszolgáló tulajdonságlapjának LDAP Routing oldalát a következő, POP3-mal foglalkozó részben mutatjuk majd be. A tulajdonságlapnak így már csak egy ismeretlen oldala maradt, ez pedig a szolgáltatás adminisztratív felügyeletének hozzáférési jogait definiáló Security fül. Itt azokat a felhasználókat sorolhatjuk fel, akik jogosultak az SMTP rendszerszolgáltatás beállításainak módosítására *(Ők alapértelmezésben természetesen az Administrators csoport, valamint a rendszerszolgáltatások által használ Local Service illetve Network Service felhasználói fiókok.).*

A kiszolgáló könyvtárstruktúrája

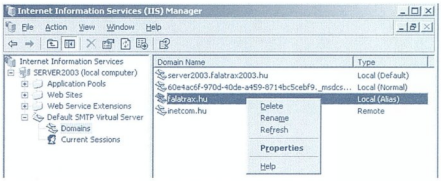
Az SMTP szolgáltatás által telepítéskor létrehozott könyvtárstruktúra alapértelmezett helye a rendszerpartíció \Inetpub\mailroot mappája. Ezen a könyvtáron belül az SMTP szolgáltatás az alábbi alkönyvtárakat hozza létre:

- **Badmail:** erről már volt szó, ide kerülnek a kézbesíthetetlen üzenetek *(NDR-ek).*
- **Drop:** a virtuális kiszolgáló alapértelmezett *(Default)* tartományának célkönyvtára; minden beérkező levél *(válogatás nélkül)* ebbe a mappába kerül be. *(Ha a POP3 szolgáltatást is telepítettük, a POP3 által kezelt tartományokba érkező levelek saját könyvtárakba kerülnek, de erről majd legközelebb.)*
- **Pickup:** az SMTP szolgáltatás folyamatosan figyeli ennek a könyvtárnak a tartalmát, és ha itt olyan fájlt talál, amely tartalmilag megfelel egy RFC 822 *(azaz SMTP)* e-mailnek, kézbesíti azt. Így szinte bármely program képes levélküldésre, ha alkalmas arra, hogy a merevlemezen szöveges állományt hozzon létre.
- **Queue:** az SMTP által kiküldésre váró üzenetek *(sikertelen küldési kísérlet esetén is itt maradnak meg, míg az újrapróbálkozási időtartam le nem telik)*
- **SortTemp:** átmeneti könyvtár, amelyet az SMTP kiszolgáló levélküldéskor használ

A virtuális kiszolgáló gyökérmappájának elérési útját a virtuális kiszolgáló létrehozásakor adhatjuk meg *(később csak a MetaBase szerkesztésével módosíthatjuk).*

A szolgáltatás által kezelt tartományok *(Domains)*

Egy SMTP virtuális kiszolgáló egynél több internetes tartomány *(domain)* üzeneteit is képes kezelni. Ezeket a tartományokat az Internet Services Manager MMC konzolfa Domains ága alatt láthatjuk:



A virtuális kiszolgáló által kezelt tartományok

A virtuális kiszolgáló alatt létrehozható tartományok két fő csoportba tartozhatnak:

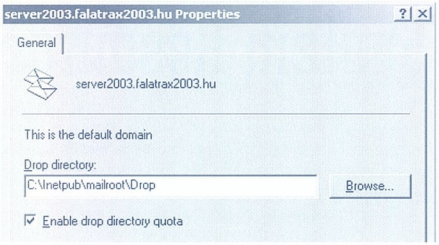
- **Local:** Helyi tartomány; minden ilyen tartomány felhasználóinak címzett levelet a virtuális kiszolgáló elfogad és kézbesít *(elhelyezi azt a Drop könyvtárban)*
- **Remote:** Távoli tartomány; ezeket a bejegyzéseket két okból hozhatjuk létre. Az egyik az, hogy így adhatjuk meg egy-egy adott tartomány SMTP kiszolgálóival felépítendő kapcsolat alapértelmezéstől különböző beállításait, a másik pedig, amikor az SMTP kiszolgáló „vállalja" hogy átmenetileg tárolja az idegen tartomány felhasználói címére érkező leveleket, hogy aztán az illetékes SMTP kiszolgáló innen tölthesse le azokat.

A helyi tartományok között több altípust is találunk:

- **Default:** az alapértelmezett helyi tartomány. Minden SMTP virtuális kiszolgáló rendelkezik legalább egy tartománnyal, ez pedig a Local *(Default).* Az alapértelmezett tartomány nem törölhető.
- **Alias:** az alapértelmezett helyi tartomány „másik" neve. A kiszolgáló fogadja az ebbe a tartományba érkező leveleket, és pontosan úgy jár el, mintha azok az alapértelmezett tartományba érkeztek volna *(beleértve a kézbesítést az alapértelmezett tartomány Drop könyvtárába).* Éppen ezért az Alias tartományok tulajdonságlapja üres.
- **Normal:** Ilyen típusú tartományt akkor láthatunk, amikor az SMTP kiszolgáló tartományvezérlőn fut. Ez a *(csúnya, GUID-nevű)* tartomány felelős az Active Directory SMTP alapú replikációs forgalmának fogadásáért és továbbításáért.
- **Custom:** Egyéni tartomány, a POP3 kiszolgáló hozza létre.

A helyi tartományok beállításai

A helyi tartományok tulajdonságlapja a következőképpen fest *(kivéve az Alias tartományokét, mert az üres):*



Az alapértelmezett helyi tartomány beállításai

Itt meghatározhatjuk a Drop könyvtár helyét *(emlékezzünk: ide kerülnek a beérkező levelek)*, illetve bekapcsolhatjuk a könyvtár méretének korlátozását *(„Enable drop directory quota")*. Ha ez az opció engedélyezve van *(és alapértelmezésben így van)*, a Drop könyvtár mérete nem haladhatja meg az SMTP kiszolgáló beállításaiban található Limit Message size to érték tízszeresét *(azaz alapértelmezésben 20 megabájtot)*. Ellenkező esetben az SMTP kiszolgáló a postaláda telítésére hivatkozva visszautasítja a beérkező leveleket.

A távoli tartományok beállításai

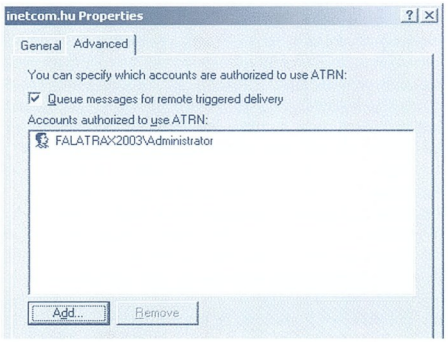
A távoli tartományok tulajdonságlapja a fentieknél jóval bőbeszédűbb:

Egy távoli tartomány tulajdonságlapja

Az opciók a következők:

- **Allow incoming mail to be relayed to this domain:** Az SMTP szolgáltatás elfogadja az ebbe a tartományba címzett bejövő leveleket és továbbítja azt a megfelelő kiszolgálónak *(azaz erre az egy tartományra nézve engedélyezzük a Mail Relay-t)*
- **Send HELO instead of EHLO:** ha bekapcsoljuk, az SMTP szolgáltatás a tartomány kiszolgálóival a kibővített SMTP helyett az „eredeti" SMTP protokollt használja *(ez jóval kevesebb szolgáltatást biztosít)*
- **Outbound Security:** A tartomány kiszolgálóihoz történő kapcsolódás biztonsági beállításai *(azonosítási mód [Anonymous/Basic/Windows Integrated], felhasználónév, jelszó, TLS beállítások)*
- **Route domain:** A levelek továbbításának módja, a szokásos DNS-alapú kiszolgálókereséssel ***(Use DNS to route this domain)*** vagy közvetlenül egy adott kiszolgálónak ***(Forward all mail to smart host)***. Ekkor meg kell adnunk a kiszolgáló DNS-nevét, vagy **[IP-címét]**.

Az SMTP szolgáltatás képes arra, hogy egy távoli tartományba címzett leveleket átmenetileg tároljon és ne próbálja meg azokat azonnal továbbítani. Erre akkor lehet szükség, ha a távoli tartomány SMTP kiszolgálója nem rendelkezik állandó internetes kapcsolattal. Amikor a távoli kiszolgáló bejelentkezik a hálózatba, az SMTP protokoll ATRN parancsa segítségével utasíthatja a kiszolgálónkat az addig gyűjtögetett és tárolt levelek továbbítására *(természetesen az előbb tárgyalt **Route domain** mező beállításainak értelmében)*.



Az ATRN opció beállítása

Ehhez a távoli tartomány tulajdonságlapjának Advanced oldalán kapcsoljuk be a **Queue messages for remote triggered delivery** opciót, majd adjunk meg egy felhasználót. A távoli kiszolgáló ezzel a felhasználónévvel és természetesen a hozzá tartozó jelszóval kezdeményezheti az üzenetküldést.

Joker-karakter a távoli tartomány nevében

Ha akarjuk, a távoli tartomány nevében használhatjuk a * *(csillag)* karaktert. Az SMTP kiszolgáló ekkor a tartomány és minden altartománya esetén a beállításoknak megfelelően fog eljárni:

```
*.remotedomain.hu
```

A következő számban a POP3 szolgáltatás bemutatásával…

…folytatjuk!

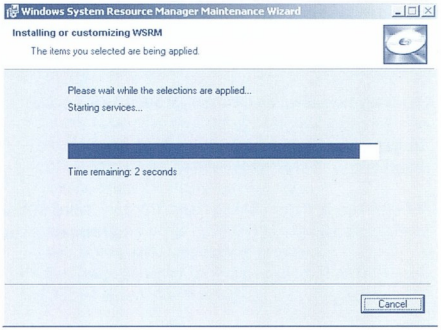
Fülöp Miklós
mick@inetcom.hu

A Windows System Resource Manager

Vége az erőforrásokért folyó versengésnek!

A Windows System Resource Manager a folyamatok prioritását dinamikusan változtató algoritmus segítségével képes arra, hogy azok erőforrás-felhasználását az előre beállított határértékek alatt tartsa, így biztosítva az erőforrások optimális kihasználtságát.

Ha több alkalmazást futtatunk egy kiszolgálón, megindul a versengés a rendszererőforrásokhoz való hozzáférésért. Ha a rendszer nem képes arra, hogy megvalósítsa az erőforrások optimális elosztását, vagy az erőforrások kihasználtsága nem lesz megfelelő, vagy a szűkös erőforrások a rendszer egészének teljesítményét fogják csökkenteni. Hogy elkerülhessük a versengést, és a valós igények szerint oszthassuk meg az erőforrásokat, erőforrás-felügyeleti szoftvert kell használnunk.
Az erőforrás-felügyeleti szoftverek használatával növelhető a kiszolgálók kihasználtsága *(a teljesítmény csökkenése nélkül)*, mivel segítségükkel a rendszergazdák meghatározhatják, hogy az egyes alkalmazások, folyamatok, vagy szolgáltatások milyen mértékben használhatják a rendszer erőforrásait. Például, ha két CPU-igényes alkalmazást futtatunk párhuzamosan, előfordulhat, hogy egyikük az igényeihez képest aránytalanul kevés CPU időhöz jut. Az erőforrásfelügyelet megoldást nyújt az ilyenfajta problémákra.
A Microsoft új erőforrás-felügyeleti eszköze a Microsoft Windows System Resource Manager *(WSRM)*, amely a Windows Server 2003 Enterprise és Datacenter változatainak része, és csak ezekben használható. Az operációs rendszer telepítése után azonban hiába keressük a WSRM bármiféle nyomát, az eszközt a csomagban kapott külön CD-ről utólag kell telepíteni. A telepítőt tartalmazó CD image az **[WSRM]** címről is letölthető.



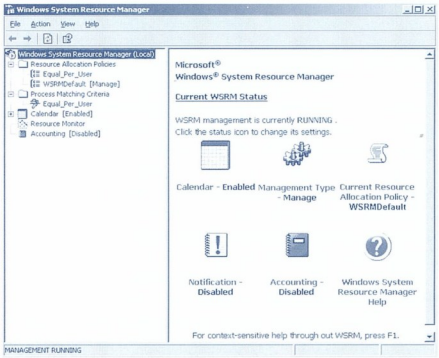
A WSRM telepítése

A Windows System Resource Manager

A WSRM lehetővé teszi, hogy a rendszergazdák meghatározzák a CPU idő és a memória megosztását az egyes alkalmazások, folyamatok és szolgáltatások között. Az eszköz lehetővé teszi a Microsoft Terminal Services-hez csatlakozó felhasználók közötti erőforrás-megosztás felügyeletét is. A WSRM teljes funkcionalitása elérhető grafikus felületről *(mmc snap-in)* és parancssori interfész segítségével is. A parancssori interfész lehetővé teszi szkriptek használatát is.
A konzol indítására szolgáló parancsikont a felügyeleti eszközök között találjuk, illetve használhatjuk a

```
wsrm.msc
```

parancsot is.
A parancssori felületet a wsrmc.exe, és annak számtalan kapcsolója biztosítja.



A Windows System Resource Manager grafikus felhasználói felülete

A CPU idő kiosztása

A WSRM által felügyelt környezetben a folyamatok két csoportba tartozhatnak: felügyelt és felügyelet nélküli folyamatok. A felügyelet nélküli folyamatok *(ezek tipikusan a rendszerfolyamatok)*, nem tartoznak a WSRM hatáskörébe, ők rendelkeznek a legmagasabb prioritással, annyi erőforráshoz jut-

hatnak, amennyi csak rendelkezésre áll. A WSRM csak a felügyelet nélküli folyamatok által fel nem használt erőforrások felett rendelkezik. Ez az oka annak, hogy a felügyelt folyamatok között az összes rendelkezésre állónál kevesebb erőforrást oszthatunk szét.

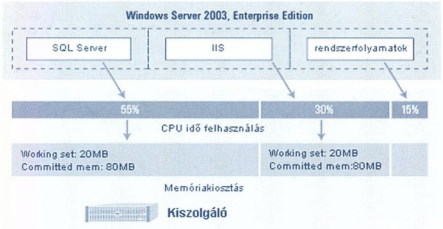
A WSRM használatával a CPU idő elosztására alapvetően két módszer áll rendelkezésre. Egyrészt meghatározhatjuk, hogy a folyamat a rendelkezésre álló teljes CPU idő *(az összes processzorra vonatkozóan)* hány százalékát használhatja fel, másrészt megadható az is, hogy a folyamatok melyik processzoron *(vagy processzorokon)* futhatnak *(processor affininty)*.

A definiált korlátozások szigorúan érvényre jutnak, ha több folyamat verseng a CPU időért, de ha másik folyamat nem igényli azt, ami járna neki, a felügyelt folyamat többet is felhasználhat a rendszergazda által meghatározott mennyiségnél. Ez a rugalmasság különösen előnyös akkor, ha a rendszerben futó alkalmazások terhelése nem egyenletes. Ilyenkor az erőforrások elosztását az egyes alkalmazások átlagos terhelése alapján végezhetjük el, de az aktuálisan nagyobb terheléssel működő alkalmazás elveheti a többiektől az éppen nem használt erőforrásokat. Ilyen módon az erőforrások kihasználtsága folyamatosan optimális lehet.

A memória kiosztása

A WSRM lehetővé teszi, hogy a folyamatok memóriahasználatát két szempont szerint korlátozzuk: beállíthatjuk a folyamat által felhasználható fizikai memória *(working set)*, és a folyamat számára kiosztott összes memória *(committed memory)* maximális mennyiségét. A working set a folyamat számára kiosztott virtuális lapok azon része, amelyek a fizikai memóriában helyezkednek el. Ha a folyamat memóriahasználata eléri a beállított korlátot, a WSRM beavatkozik, és megakadályozza, hogy a folyamat további lapokhoz jusson a fizikai memóriában, illetve kezdeményezi a working set egy részének kilapozását.

A committed memory a fizikai memóriának az a része, amelynek a memóriakezelő lefoglalta a megfelelő helyet a háttértáron lévő lapozófájlban arra az esetre, ha a lapokat a lemezre kellene írnia. Memóriakezelési problémára *(memóriaszivárgás)* utal, ha egy folyamat növekvő mértékben használ committed memóriát, ekkor valószínűleg nem szabadítja fel az előzőleg lefoglalt, de már nem használt lapokat. Ha a folyamat memóriahasználata eléri a beállított értéket, a WSRM választásunk szerint leállítja azt, illetve hibajelzést ír a rendszernaplóba.



A WSRM az erőforráselosztási házirend beállításai alapján felügyeli az erőforrásokhoz való hozzáférést

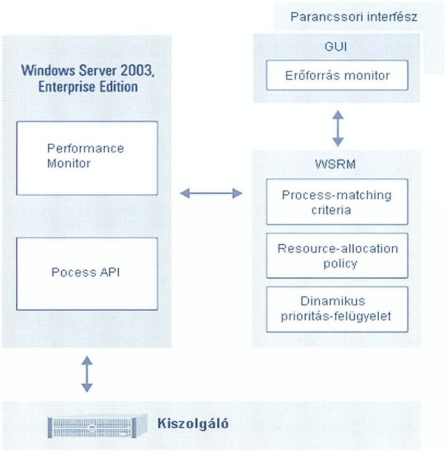
A WSRM felépítése

A WSRM a rendszererőforrások felügyeletét a folyamatok prioritását dinamikusan változtató algoritmus segítségével végzi, amely meghatározza az erőforrások elosztását a felügyelt folyamatok között. A WSRM meghatározott lekérdezési ciklus szerint ellenőrzi a felügyelet folyamatok által lefoglalt erőforrások mennyiségét. Minden lekérdezés alkalmával összehasonlítja az aktuális erőforrás-felhasználást a beállított értékekkel. Ha egy folyamat túl sok erőforrást foglal, a WSRM csökkenti annak prioritását.

Minden folyamat egy vagy több végrehajtási szálból áll, ezek a folyamat végrehajtható komponensei. Mivel a szülőfolyamat prioritási szintje nagymértékben befolyásolja szálainak prioritását, a folyamat prioritásának csökkentése a szálak prioritását is csökkenti. A Windows operációs rendszer prioritásos preemptív *(round-robin)* ütemezési rendszert használ, amely a szálakat prioritásuknak megfelelő sorrendben hajtja végre. Ilyen módon a folyamat prioritásának csökkentése, indirekt módon egyben az által felhasznált erőforrások mennyiségét is csökkenti.

Az alkalmazások, folyamatok és szolgáltatások erőforrás-felhasználásának beállítását két lépésben kell elvégeznünk:

- meghatározzuk a folyamatoknak azon csoportját, amelyekre a beállítandó szabály vonatkozni fog *(process-matching criteria)*
- az adott folyamatok számára meghatározzuk a felhasználható erőforrások maximális mennyiségét *(resource-allocation policies)*



A WSRM felépítése

A felügyelt folyamatok kiválasztása

A WSRM lehetőséget nyújt arra, hogy a felügyelni kívánt folyamatokat csoportokba rendezzük, és a létrehozandó erőforrás-felhasználási szabályokat ezekhez a csoportokhoz rendeljük hozzá. Ha a csoporthoz CPU korlátozást állítunk be, a WSRM egyenletesen osztja el a megadott CPU időt a csoporthoz tartozó folyamatok között. Azonban ha memóriahasználatra vonatkozó korlátozást adunk meg, a korlát minden egyes folyamatra külön-külön értelmezendő. A csoporthoz tartozó

folyamatok kiválasztására szolgáló szempontokat *(process-matching criteria)* az mmc konzol felületén és parancssorból is megadhatjuk. A WSRM a folyamatok következő három tulajdonságát hasonlítja össze a beállított értékekkel:

- a folyamatot létrehozó alkalmazás végrehajtható fájljának teljes útvonala, illetve annak bármilyen része.
- a folyamat létrehozásának és elindításának pillanatában átvett parancssori sztring bármilyen része.
- a folyamatot elindító felhasználó, vagy felhasználói csoport.

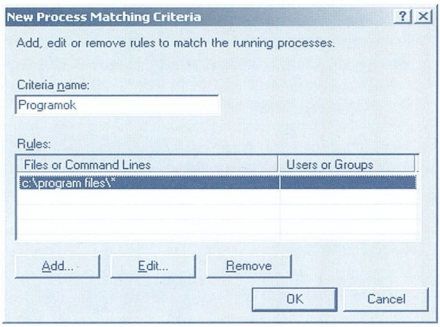
Minden olyan folyamat, amely egyetlen beállított érték szerint sem azonosítható, az alapértelmezett csoport *(default group)* tagja lesz.

A fájlnév, illetve útvonal megadásakor helyettesítő karaktereket is használhatunk, ha egyszerre több alkalmazást is szeretnénk kiválasztani.

Például a

```
c:\program files\*
```

megadása az összes olyan folyamatot kijelöli, aminek teljes útvonalában a "c:\program files" sztring előfordul.



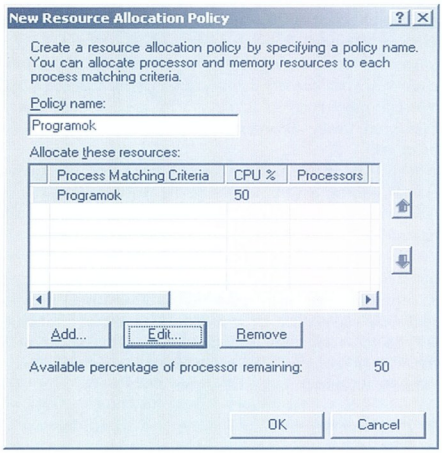
A csoporthoz tartozó programok kiválasztása

A kritikus rendszerfolyamatok alapértelmezés szerint felügyelet nélküliek, így ezek bármennyit felhasználhatnak a rendelkezésre álló rendszererőforrásokból. Ennek megvalósítása kivétellisták *(exclusion lists)* segítségével történik, az ezeken szereplő folyamatok mindegyike felügyelet nélküli. Két kivétellistánk is van, mindkettő tartalma megtekinthető, ha a konzolfában megnyitjuk a Windows System Resource Manager tulajdonságlapját. A rendszer által meghatározott *(system-defined)* lista tartalma nem módosítható, ezen szerepelnek a legfontosabb rendszerfolyamatok. A felhasználó által meghatározott *(user-defined)* listát azonban tetszés szerint bővíthetjük, illetve szűkíthetjük. A kivétellistákon szereplő valamennyi folyamat felügyelet nélküli lesz.

Az erőforrás-elosztási házirendek

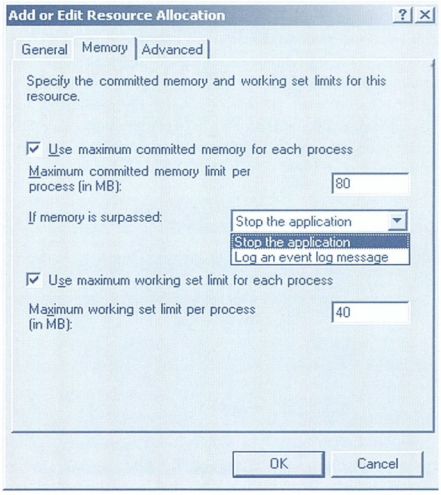
A felügyelt folyamatokhoz az erőforrás-elosztási házirendek *(resource-allocation policies)* segítségével rendelhetjük hozzá a nekik szánt erőforrásokat. Minden erőforrás-elosztási házirend egy vagy több folyamatcsoportot tartalmaz, amelyekre vonatkozóan megadhatjuk a kívánt CPU felhasználási szintet, a memóriára vonatkozó korlátozásokat és a processzoraffinitással kapcsolatos adatokat.

Az alábbi ábrán látható, hogy a létrehozandó erőforrás- elosztási házirendhez hozzárendeljük a korábban létrehozott csoportot *(amely a c:\program files mappában lévő összes programot tartalmazza)* és nekiadjuk a processzoridő 50 százalékát.



Erőforráselosztási házirend létrehozása

Miután elvégeztük a csoport és a korlátozások összerendelését, a WSRM figyelni kezdi a csoporthoz tartozó folyamatok erőforrás-felhasználását. Ha a CPU használat meghaladja a beállított értéket, a WSRM csökkenti a folyamatok prioritását, így a Windows ütemezőjétől az kevesebb processzoridőt fog kapni. Korábban már említettük, és az alábbi ábrán is látható *(1)*, hogy a CPU idővel ellentétben a felhasználható memória korlátozása a csoport minden egyes folyamatára külön-külön értendő.



A memória kiosztása

Minden erőforráselosztási házirendhez automatikusan létrejön egy alapértelmezett csoport *(default group)*, amely tartalmazza az összes olyan folyamatot, amelyek nem feleltek meg egyetlen kiválasztási feltételnek, és nem szerepelnek a kivétellistákon sem. Az alapértelmezett csoport tagjai egyenlően osztoznak azokon az erőforrásokon, amelyeket a felügyelet nélküli és a felügyelt folyamatok még nem használtak fel.

A WSRM Calendar

A szolgáltatás lehetővé teszi az erőforrás-elosztási házirendek előre megadott időzítés szerinti, automatikus aktiválását. Erre akkor van szükség, ha a felügyelt alkalmazások terhelése, és ezzel párhuzamosan erőforrásigénye egy adott időszak alatt szisztematikusan változik.
A szolgáltatás segítségével három különböző időzítéstípus hozható létre:

- egyszeri esemény *(OneTime Event)*
- rendszeres esemény *(Recurring Event)*
- napi időrend *(24 órás periódus alatt beállítható több erőforrás-elosztási házirend aktiválása, majd a napi időrend hozzárendelhető a megfelelő naptári napokhoz, Schedule)*

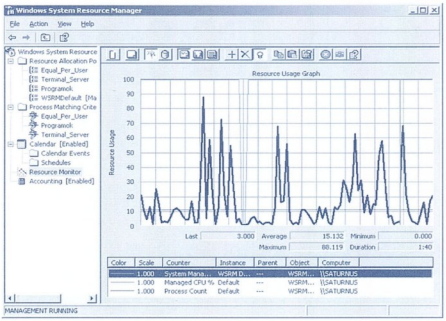
A teljesítményadatok vizsgálata a Resource Monitor segítségével

A Resource Monitor GUI segítségével megtekinthetjük a helyi gép, illetve távoli számítógépek real-time teljesítményadatait. A Resource Monitor szabványos Windows rendszermonitor *(PerfMon)*, amelyhez három új teljesítményszámláló tartozik:

- WSRM: Policy
- WSRM: Process
- WSRM: Process Matching Criteria

A Resource Monitor által megjelenített adatokat a következő paraméterek megadásával definiálhatjuk:

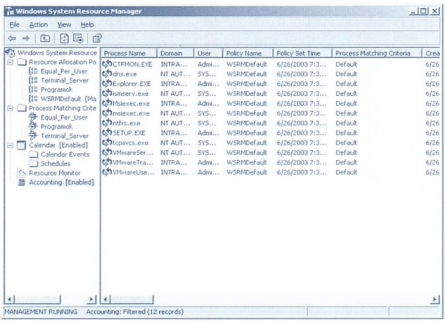
- Az adatok típusa. Meg kell határoznunk az objektumot *(például folyamatok, vagy processzor)* és ki kell választanunk az objektumhoz tartozó teljesítményszámlálók közül a megfelelőt *(például working set mérete, vagy % Processor time)*.
- Az adatok forrása. A Resource Monitor képes a helyi gépen kívül, a hálózatban elérhető számítógépek adatainak megjelenítésére is. Az adatok lekérdezéséhez alapértelmezés szerint rendszergazdai jogosultság szükséges. Megjeleníthetők továbbá a korábban számlálónaplókba *(counter logs)* gyűjtött adatok is.
- Mintavételi paraméterek. A Resource Monitor támogatja az igény szerinti manuális, és a megadott időközönkénti automatikus mintavételezést is a real-time adatok esetében. A tárolt adatok megjelenítésekor a kezdő és befejező időpontot adhatjuk meg, így megjeleníthetők a vizsgált időszak adatai.



A Resource Monitor

A WSRM accounting

A szolgáltatás segítségével eltárolhatjuk és visszatölthetjük a felügyelt folyamatok viselkedésével kapcsolatos adatokat. Az adatokat felhasználhatjuk jelentések készítéséhez, és a felügyelt folyamatok erőforrás felhasználással kapcsolatos viselkedésének tanulmányozásához.



A WSRM accounting

A kiszolgálók kihasználtságának növekedése

Ha több alkalmazást futtatunk egy kiszolgálón, a WSRM használatával jelentősen növekedhet a kiszolgáló kihasználtsága, mivel segítségével a rendszergazdák kontrollálhatják az egyes alkalmazások, folyamatok, és szolgáltatások által felhasznált erőforrások mennyiségét. Ilyen módon megelőzhető az erőforrásokért való versengés, és optimalizálható a szűkös erőforrások elosztása és kihasználtsága.

Szerényi László
szerenyi.l@met.hu

ISA Server 2000, Feature Pack 1

Új téglák a (tűz)falban

A nemrégiben megjelent ISA Server 2000 Feature Pack 1 segítségével tűzfalunk funkciói kiteljesednek *(kiterjedt SMTP-filter, URLScan 2.5 stb.)*, néhány mindennapos feladat megoldása sokkal egyszerűbbé válik *(például az OWA kiszolgálóik publikálása)*. És ez még nem minden!

Áttekintés

A Microsoft Internet Security and Acceleration Server 2000 Feature Pack 1 a szokványos tűzfalaknál nagyobb biztonságot nyújt a Microsoft Exchange és IIS kiszolgálók üzemeltetői számára. A vállalatok egyre nagyobb mértékben használják üzleti folyamataikban az e-mailt és a Web lehetőségeit, így egyre nagyobb igény mutatkozik részükről az e-mail és Web kiszolgálók fokozottabb védelmére. Az ISA 2000 Feature Pack 1 segítségével a nagyobb biztonság egyszerűbb használattal és felügyelettel párosul.

Az ISA Server 2000 Feature Pack 1 a következő fejlesztéseket tartalmazza:

- Továbbfejlesztett SMTP szűrő.
- Továbbfejlesztett Exchange távoli eljáráshívás *(RPC)* szűrő.
- URLScan 2.5 for ISA Server
- RSA SecureID hitelesítés használata
- A hagyományos és a SecureID hitelesítés delegálása
- Outlook Web Access Wizard
- RPC Filter Configuration Wizard
- Link Translator
- A felhasználással és hibaelhárítással kapcsolatos dokumentáció.

A továbbiakban részletesen áttekintjük az említett új lehetőségeket, amelyek segítségével védettebbé tehetők az e-mail, Web, és OWA kiszolgálók, és egyszerűbbé válhat a rendszerfelügyelet. Az ISA Server 2000 Feature Pack 1 az **[ISAFP]** címről tölthető le.

Az e-mail kiszolgálók védelme

Az ISA Server 2000 Feature Pack 1 a következő szolgáltatások segítségével fokozza az e-mail kiszolgálók biztonsági szintjét:

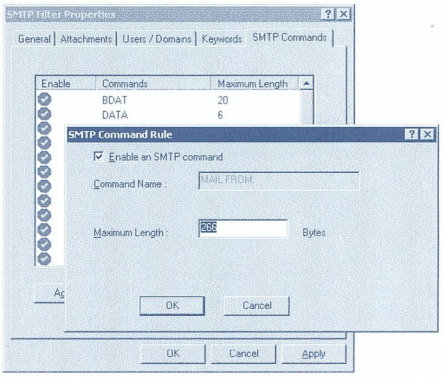
- Az SMTP szűrő továbbfejlesztett változata nagyobb védelmet nyújt a nem kívánt e-mail üzenetekkel szemben.
- VPN nélkül is védelmet biztosít a távoli Outlook kliensek számára.

Az ISA kiszolgáló alkalmazásszintű szűrés segítségével teremti meg az e-mail kiszolgálók biztonságát. Az e-mail forgalom folyamatos figyelése csökkenti annak esélyét, hogy vírusok és nem kívánt üzenetek kerüljenek be a vállalati hálózatba.

Az SMTP szűrő feladata az, hogy megvizsgálja a 25-ös portra érkező teljes forgalmat. A szűrő fogadja és feldolgozza az üzeneteket, és a beállított szabályoknak megfelelően továbbítja, illetve blokkolja azokat. A szűrés történhet a mellékletek neve, kiterjesztése és mérete alapján, valamint a feladó, a domain, bármely kulcsszó, vagy SMTP parancs, illetve annak hossza szerint is.

Az üzenetek szűrését két különállóan implementált komponens valósítja meg:

- Az SMTP szűrő az engedélyezett, illetve tiltott SMTP parancsok szűrését valósítja meg, az üzenetek tartalmával NEM foglalkozik. Az SMTP szűrő alapértelmezés szerint fel van telepítve, de működését külön engedélyeznünk kell.
- Az üzenetfigyelő *(message screener)* az Exchange 2000 SMTP kiszolgálójának egyik kiterjesztése, amely opcionálisan telepíthető. Az ISA kiszolgáló telepítése közben csak akkor kérhetjük az üzenetfigyelő telepítését, ha az SMTP kiszolgáló már telepítve van a számítógépen. Az üzenetfigyelő feladata az üzenetek tartalmának szűrése a megadott kulcsszavak, melléklet-típusok, és feladók szerint. Az SMTP szűrő az üzenetfigyelő nélkül is telepíthető, de ebben az esetben csak az SMTP parancsok szűrésére van lehetőség. Az üzenetfigyelő beállításait az SMTP szűrő tulajdonságlapján adhatjuk meg.

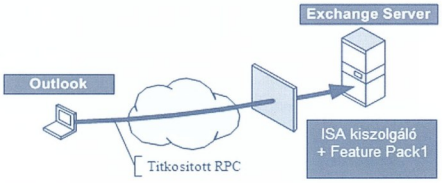


Az SMTP szűrő képes (sok egyéb mellett) bármely SMTP parancs hosszát is figyelembe véve válogatni a levelek között

Az ISA kiszolgáló lehetővé teszi, hogy az Outlook kliensek és az Exchange kiszolgálók közötti e-mail forgalom megbízhatatlan hálózatokon keresztül is VPN használata nélkül bonyolódhasson. Ezt a képességet fejleszti tovább a Feature Pack 1 a következő tulajdonságok segítségével:

Kötelező RPC titkosítás

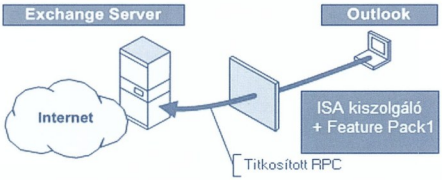
Az ISA 2000 Feature Pack 1 lehetővé teszi, hogy a rendszergazdák kötelezővé tegyék a távoli eljáráshívással *(RPC)* kapcsolatos kommunikáció titkosítását az Outlook kliensek és az Exchange kiszolgálók között. Így VPN kapcsolat nélkül is folyhat titkosított kommunikáció.



Titkosított RPC az Outlook kliensek és az Exchange kiszolgáló között

Kimenő RPC kommunikáció engedélyezése

Az ISA 2000 Feature Pack 1 lehetővé teszi, hogy az ISA által védett Outlook kliensek RPC kommunikációt folytassanak a tűzfalon kívül elhelyezkedő Exchange kiszolgálókkal.



A Feature Pack 1 lehetővé teszi a kimenő RPC kommunikációt

A Web és OWA kiszolgálók védelme

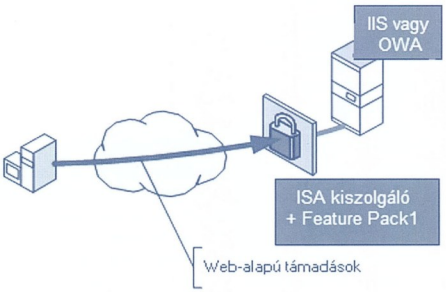
Az ISA 2000 Feature Pack 1 a következő fejlesztések segítségével fokozza a Web és OWA kiszolgálók biztonságát:

- Védelmet nyújt az Interneten lévő kiszolgálók elleni támadások legújabb típusaival szemben is.
- Az hitelesítés új módszereinek segítségével a hozzáférés jobban kézbentartható.

Az alkalmazások egyre nagyobb része használja kommunikációs célra a HTTP és a HTTPS protokollokat, így egyre több, ezeken a protokollokon utazó féreg és vírus támadja meg a vállalati hálózatokat. Sajnos a hagyományos csomagszűrő tűzfalak képtelenek feltartóztatni az ilyen típusú támadásokat, ehhez az alkalmazási rétegben működő tűzfalra van szükség. Az ISA Server 2000 és a Feature Pack 1 megkönnyíti a biztonsági beállítások felügyeletét, és segítségükkel a kártékony Web kérések már a tűzfal számítógépen blokkolhatók, így nem juthatnak be a vállalat hálózatába.

URLScan 2.5 for ISA Server

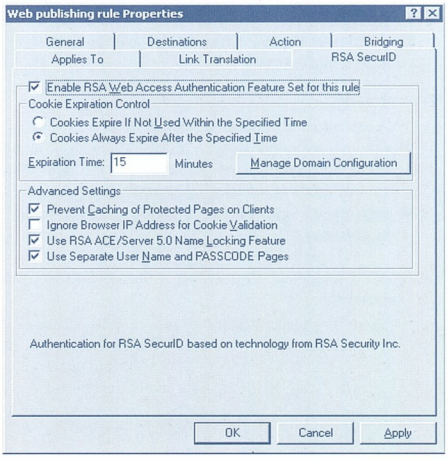
A webalapú támadások általában szokatlan kéréseket tartalmaznak, hosszú karakterláncokból állnak, vagy különleges karakterkészlet szerint vannak kódolva. Az URLScan teszi lehetővé, hogy az ISA kiszolgálók észleljék és elhárítsák az ilyen típusú támadásokat. Ha az URLScan nem minden egyes web és OWA kiszolgálón, hanem csak a védelmüket ellátó ISA kiszolgálón működik, a rendszer komplexitása, és a felügyelet munkaigénye is jelentősen csökkenthető.



Az IIS és OWA kiszolgálókat a tűzfalon futó URLScan védi

RSA SecurID hitelesítés

Az RSA SecureID hitelesítés segítségével korlátozható a Web és OWA kiszolgálók felé irányuló forgalom; a vállalat hálózatba csak érvényes kérések juthatnak be.



A SecurID hitelesítés engedélyezése az ISA kiszolgálón

Ha a felhasználó SecurID-vel védett weblapok elérését kezdeményezi, az ISA Servert futtató kiszolgáló *(a védett kiszolgáló nevében)* bekéri a felhasználó nevét és PASSCODE-ját. Az ISA kiszolgálón futó RSA ACE/Agent továbbadja az adatokat az

RSA ACE/Server-nek. Ha az azonosítás sikeresen megtörtént, a felhasználó böngészője egy cookie formájában megkapja a felhatalmazást, hogy az adott munkamenet tartama alatt a védett tartalomhoz hozzáférjen.

Az RSA SecurID hitelesítés használatának beállítása a következő lépésekből áll:

- Az RSA ACE/Serveren az ISA kiszolgáló megadása, mint RSA ACE/Agent
- A felhasználók hozzáadása az RSA ACE/ Agent Host recordhoz
- Az ACE/Serverrel való kapcsolat tesztelése
- Webpublikálási szabály létrehozása
- Az ügyfél és az ISA kiszolgáló közötti biztonságos kapcsolat beállítása.

A hagyományos és a SecurID hitelesítés delegálása

A Feature Pack 1 segítségével lehetővé válik az is, hogy az RSA SecurID alapú, illetve a hagyományos hitelesítést is az ISA kiszolgáló végezze el a Web és OWA kiszolgálók számára. A védett kiszolgálók az ISA Server-re bízzák az ügyfelek hitelesítését, így nincsen szükség arra, hogy a felhasználók többször megadják hitelesítési adataikat. A delegáció minden egyes Web publikálási szabály esetében külön engedélyezhető, illetve tiltható.

Könnyebb használat és rendszerfelügyelet

Az ISA Feature Pack 1 a következő eszközöket biztosítja a rendszergazdák számára:

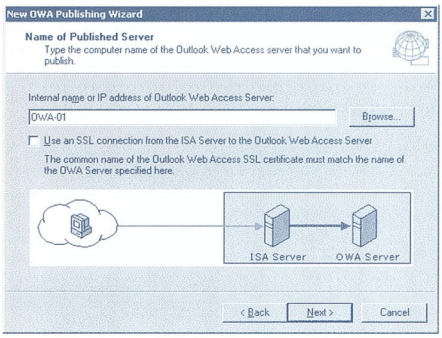
- Könnyen használható varázslók teszik egyszerűbbé a hálózat védelmét.
- A vállalati intraneten lévő információk könnyen közzétehetők az Interneten.
- A dokumentációban a gyakoribb kérdésekre gyorsan választ találhatnak.

OWA varázsló

Az OWA varázsló a következő szolgáltatásokat biztosítja:

- Automatikusan generálja a megfelelő értékeket tartalmazó Web publikációs szabályokat *(Web publishing rule)* és célcímkészleteket *(destination set)*.
- A megfelelő figyelőket hozzárendeli az ISA Server külső címeihez.

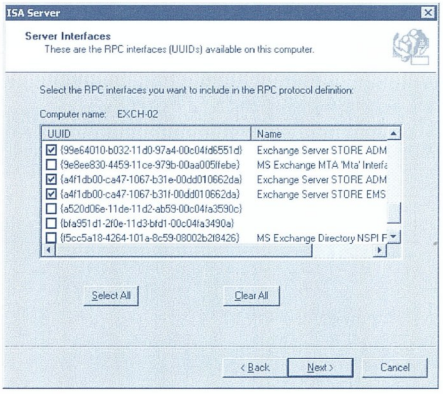
Kiválasztja a megfelelő tanúsítványt a Secure Socket Layer *(SSL)* kapcsolathoz.



Az OWA kiszolgálók védelmének gyors és egyszerű beállítása az ISA-n

RPC Filter Configuration Wizard

Az ISA Serveren eddig az RPC szolgáltatásokhoz való hozzáférést csak mindent vagy semmit alapon tudtuk befolyásolni. Az ISA 2000 Feature Pack 1 azonban lehetővé teszi, hogy meghatározzuk az egyes RPC szolgáltatásokhoz való hozzáférés lehetőségét is. Választhatunk a varázsló által megjelenített listából, illetve magunk is összeválogathatjuk az RPC interfészt. A szolgáltatásdefiníciókat a publikálási szabályban is felhasználhatjuk, így a külső ügyfelek is elérhetik azokat.



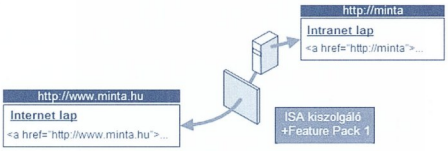
A belső RPC szolgáltatásokhoz való hozzáférés finomhangolása az RPC varázsló segítségével.

Link Translator

Az ISA Server Feature Pack 1-ben mutatkozik be a Link Translator, amely lehetővé teszi az abszolút hivatkozásokat tartalmazó webhelyek publikálását. A link translation segítségével az abszolút hivatkozásokat tartalmazó webhelyeket a külső felhasználók is hibás linkektől mentesen érhetik el. Az Internet felől érkező böngészők számára a Link Translator az ilyen hivatkozásokat átalakítja a megfelelő webes hivatkozásokká. A szolgáltatás használatával nincsen szükség az

intranet oldalak módosítására, hogy a külső felhasználók számára is elérhetővé váljanak.

Az ábrán látható példa bemutatja a Link Translator működését:



- Az Internetről érkező ügyfél lekéri a www.minta.hu webkiszolgálón található lapot.
- A kiszolgáló által tárolt lap abszolút hivatkozást tartalmaz a minta.hu intranetes számítógépére *(http://minta)*. A Link Translator használata nélkül a felhasználóhoz érkező lap a működésképtelen intranet címeket tartalmazná. A problémát csak az összes hivatkozás manuális módosításával oldhatnánk meg.
- Amikor a lap áthalad az ISA kiszolgálón működő Link Translatoron, az összes intranetes hivatkozás *(http://minta)* Internet címmé módosul *(http://www.minta.hu)*.
- A felhasználó által kapott oldalon az összes hivatkozás megfelelően működik.

A Link Translator a hivatkozások módosítását az alapértelmezett hivatkozás-fordítási szótár *(link translation dictionary)* segítségével végzi, amely létrejön minden Web publikálási szabályhoz, amelyen a hivatkozások fordítását engedélyezzük. A hivatkozás-fordítási szótár az igényeknek megfelelően bővíthető.
A Link Translator a következők szerint végzi a szótárban való keresést:

- Először a leghosszabb keresési sztring, majd csökkenő sorrendben a többi, végül az alapértelmezett sztringek.
- Találat esetén a szűrő megvizsgálja a sztringet követő karaktert
- Ha a karakter termináló karakter, a szűrő elvégzi a megadott helyettesítést. A termináló karakterek teljes listája az ISA Server Feature Pack 1 on-line súgójában található.

Tekintsük át az előző példa szerinti helyzetben a beállítás szükséges lépéseit.

A helyzet tehát a következő:
A belső webhelyet, amelynek NetBIOS neve "Minta" publikáljuk az Interneten http://www.minta.hu néven. A webhelyen a Web kiszolgáló NetBIOS nevét tartalmazó hivatkozások is vannak. Olyan linkek is lehetnek a webhelyen, amelyek a belső hálózaton lévő *(kívülről el nem érhető)* Web kiszolgálók NetBIOS neveit tartalmazzák.

- Engedélyezzük a Link Translator Web szűrőt.
- Létrehozzuk a www.minta.hu-t tartalmazó célcímkészletet.
- A kiszolgáló publikálásához létrehozzuk a megfelelő Web publikálási szabályt.
- A publikálási szabályhoz létrehozzuk a megfelelő hivatkozás-fordítási szótárt.

A Link Translator segítségével módosított hivatkozások *(amelyek a belső hálózatba mutatnak)* potenciálisan biztonsági kockázatot jelentenek. A kockázat csökkentése érdekében gondosan kell definiálnunk az ISA kiszolgáló célcímkészleteit és webpublikálási szabályait.

A felhasználással és hibaelhárítással kapcsolatos dokumentáció

Forgatókönyvek és technikai dokumentáció könnyíti meg a rendszer megfelelő beállítását, segítségükkel az Exchange és IIS telepítések gyorsan és könnyen elvégezhetők. A Feature Pack 1-hez tartozó dokumentációból minden szokásos kérdésre választ kaphatunk. A dokumentáció a következő elemeket tartalmazza:

- Az ISA 2000 Server Feature Pack 1 dokumentációja.
- Webpublikálással kapcsolatos dokumentáció.
- Az Exchange kiszolgálók publikálásával kapcsolatos dokumentáció.

Szerényi László
szerenyi.l@met.hu

Biztonságos aláírás kezelő alkalmazás készítése III.

Avagy a bürokrácia diszkrét bája

Az ember minden épeszű mértéken áthágva képes elbonyolítani tulajdon életét. Ez különösen igaz társadalmi méretekben, pláne ha az adminisztráció oldaláról közelítjük meg a jelenséget. Az elektronikus dokumentumkezelés és aláírás sajnos ezen mit sem javít: csak a hagyományos papír alapú adminisztrációt képzi le elektronikus formára. A kommunikáció felgyorsulhat és egyszerűsödhet, de a folyamatok alapvetően nem változnak. Mivel az aláírás *(akár a hagyományos, akár az elektronikus)* egy jogi szempontból is érvényes kötelezettségvállalást fejezhet ki, duplán igaz rá minden bürokratikus szabály. Nem az elektronikus aláírás bonyolult, hanem az élet, aminek próbál megfelelni.

Kötelezettségvállalások

Az előző fejezetben többször volt hivatkozás ún. kötelezettségvállalási típusokra minden magyarázat nélkül. Bizonyára többen is majd lerágták a körmüket izgalmukban, hogy mi is ez az igen érdekfeszítően hangzó dolog. De, ahogy a filmsorozatok is a legizgalmasabb résznél szakadnak meg két rész között, nálunk is itt jött az oldal vége.

A jobb megértés végett a hagyományos aláírás analógiája mentén kezdenék: egy aláírással sok mindent kifejezhetünk. Ugyanaz az aláírásunk mást és mást jelenthet a körülményektől és szándékunktól függően, mely jobb esetben kiderül a *(szöveg)*környezetből, rosszabb esetben nem. Például egy vállalkozási szerződésen szerepelhet a megrendelő beszerzési igazgatójának és jogi osztályvezetőjének, s a beszállító két ügyvezetőjének és jogászának aláírása *(lásd aláírói szerepkörök)*. Egy ingatlan adásvételen a vevő és eladó, valamint az ellenjegyző ügyvéd aláírása, egy ingójelzálog bejegyzési okiraton a zálogba adóé és a közjegyzőé. Egy házassági szerződésen a férjjelölt és feleségjelölt, egy kölcsönszerződésen a kölcsönbeadó és vevő, valamint a két tanú aláírása. Mindegyik esetben a szerepekből adódóan is más-más kötelezettségvállalás történik. Némelyik egyértelmű a szerződés szövegéből, némelyik nem *(pl. tanúk szerepe)*.

De az is lehet, hogy nem szerződésről, hanem egy megrendelésről, egy visszaigazolásról, egy ajánlatkérésről vagy ajánlatról, vagy csak egy sima érdeklődésről van szó abban a levélben, mely aláírásra kerül. Ez esetben a kötelezettségvállalás típusa máris kevésbé egyértelmű. Különösen, ha nagy szervezetekről van szó, és sokszereplős ügyletekről. Ki tudja, ki, mikor, mire vállalhat egyáltalán kötelezettséget. Ugyanaz a személy ugyanazon aláírása az egyik alkalommal kifejezhet egy teljes körű kötelezettségvállalást, máskor meg csak egy szimpla láttamozást. Papír esetében például attól, hogy az egyik a lap jobb alsó sarkában van apróban, a másik meg egy vonal és megrendelő felirat felett nagy méretben. Elektronikus dokumentumoknál az ilyen bevett trükkökre nincs mód, s nem is biztos, hogy ez lenne az üdvözítő.

A megoldás az, hogy az adott környezetre *(melyre az aláírási szabályzat vonatkozik)*, az aláírás ellenőrzési szabályzat az összes szóba jöhető kötelezettségvállalási típust definiálja. Ezek közül az aláírás megtételekor vagy azt megelőzően az aláíró választhat, mely ezáltal az aláírás részévé válik. Az aláírás ellenőrzőjének a kötelezettségvállalás típusát *(csakúgy, mint az aláírási szabályzat egyéb paramétereit)* egyértelműen fel kell ismerni, s jelentésével tisztában kell lenni.

Egy kötelezettségvállalás meghatározása egy egyedi objektumazonosítóból *(OID)* és egy minősítésből áll. A minősítés a kötelezettségvállalás bővebb *(pl. szöveges)* leírását tartalmazza. A kötelezettségvállalás típusa definiálva lehet az aláírási szabályzaton belül is, de meghatározhatja egy külső *(az aláírási szabályzat által elismert)* környezet is. Így meghatározhatja jogszabály, szabvány, szerződés, egyéb alkalmazási környezet-jellemző, valamint valamilyen definíciós tár is.

A letagadhatatlanság kezelésére például a következő típusú kötelezettségvállalások szolgálhatnak:

- **Előállítás igazolása:** Az aláíró készítette az üzenetet *(de nem feltétlenül hagyta jóvá és küldte el azt)*.
- **Jóváhagyás igazolása:** Az aláíró jóváhagyta az üzenetet *(de nem feltétlenül küldte el azt)*.
- **Küldés igazolása:** Az aláíró küldte el az üzenetet *(de nem feltétlenül ő készítette és hagyta jóvá)*.
- **Származás igazolása:** Az aláíró elismeri, hogy ő készítette, hagyta jóvá és küldte az üzenetet.
- **Kézbesítés igazolása:** A szolgáltató jelzi, hogy az üzenetet egy a címzett által hozzáférhető lokális tárolóba *(tipikusan annak postaládájába)* eljuttatta.
- **Átvétel igazolása:** Az aláíró elismeri, hogy átvette az üzenetet.

Ezek a kötelezettségvállalások egymásra épülnek *(egyre magasabb bizonyosságot jelentenek)*.

E mellett definiálni lehet az aláíró szerepkörökkel összefüggésben további kötelezettségvállalásokat is *(pl. ügyvédi, közjegyzői ellenjegyzés, láttamozás stb.)*.

A kötelezettségvállalás típusa az aláírói dokumentumból is kiderülhet explicit vagy implicit módon. Például egy *(az APEH által meghatározott formátumú)* adóbevallás típusú üzenet aláírása explicit az adatok hitelességének elismerését jelenti. Implicit módon az adott dokumentum szövegéből is következhet az aláírás jelentése.

Aláíró szerepkörök

Az aláíró szerepkörök az aláíró tevékenységét, jogosultságait, illetve egy szervezetben elfoglalt szerepét írhatják le. Az aláíró neve mellett fontos lehet annak a szervezetben *(pl. cégben)* betöltött szerepe is. Egyes szerződések csak akkor tekinthetők érvényesnek, ha egy megfelelő pozíciót betöltő fél *(pl. osztályvezető)* írta alá. Sokszor fontosabb arról meggyőződni, hogy az aláíró a szervezeten belül milyen pozíciót tölt be, mint az, hogy ki is ő valójában. E mellett egy közösségben speciális szerepe lehet egyes aláíróknak, pl. az ügyvédeknek, közjegyzőknek, különböző hatóságoknak is. A szerepkör jelzése az aláírásban történhet az aláíró állítása szerint *(kinyilvánított szerepkör)* és külsőleg hitelesített módon *(pl. attribútum tanúsítvány segítségével)*. A szerepkörök és az ezek jelzésére elvárt módszer az egyes kötelezettségvállalási típusokhoz is köthetők. Így meghatározható, hogy milyen kötelezettségek, milyen szerepkör birtokában vállalhatók.
Az aláírási szabályzatnak, amennyiben definiál aláíró szerepeket, azt egyértelműen megkülönböztethető értékek hozzárendelésével kell tennie, különösen, ha dinamikus, gépileg feldolgozható formában teszi.

Algoritmusok

Az aláíró algoritmusok közé a nyilvános kulcsú rejtjelező algoritmusok mellett beleértendők a lenyomatképző és feltöltő algoritmusok is, s ezek a minimális kulcshosszra vonatkozó előírással együtt egyaránt meghatározhatják:

- Az aláíró által az aláírás során, és ellenőrző által az aláírás ellenőrzésekor felhasználható eljárását
- A végfelhasználói tanúsítványok aláírására és ellenőrzésére szolgáló algoritmusokat
- A szolgáltatói tanúsítványok aláírására és ellenőrzésére szolgáló algoritmusokat
- A visszavonási listák aláírására és ellenőrzésére szolgáló algoritmusokat
- Az attribútum tanúsítványok aláírására és ellenőrzésére szolgáló algoritmusokat
- Az időbélyegző előállítására, és ellenőrzésére felhasználható algoritmusokat

Az egyes esetekben alkalmazandó algoritmusokat az aláírási szabályzatban tisztázni kell.

Együttes és többszörös aláírások

Az együttes aláírás több dokumentum ellátását jelenti egyetlen aláírással. Erre például egy szerződés és annak különálló mellékletei esetében lehet szükség, vagy egy olyan esetben, amikor külön nyilatkozatok kapcsolódnak a szerződéshez. Papír esetében ezeket össze szokás tűzni, majd egy vagy több aláírással látják el a dokumentumot. Mivel az elektronikus aláírás a gémkapocs szerepét is átveszi *(a sértetlenség biztosításával)* az elektronikus dokumentumoknál, ilyen esetben mindenképp az egyetlen együttes aláírás megtétele javasolt.
A többszörös aláírások szerepe ennek pont a fordítottja: itt egyetlen elektronikus dokumentum van, amit több, különböző aláírással kell ellátni. Ezt akkor alkalmazzák, ha a dokumentum fontossága több személy együttes aláírását igényli. Például vannak cégek, ahol csak két felső vezető együttes aláírása számít cégszerű aláírásnak, vagy ahol bizonyos összeg felett, a bank két személy aláírását várja el a pénzügyi tranzakciókhoz. Ezen aláírások között aztán a legkülönbözőbb kapcsolatok lehetnek az alkalmazott ügylettől és a szervezeti előírásoktól függően. Alá, fölé és mellérendeltségi viszonyok, sorrendiségek lehetnek kialakítva a vonatkozó szabályokban *(pl. szervezeti és működési szabályzat)*. A céges példálózások helyett vegyünk most egy pártot: a párt elnöke csak akkor fog aláírni bármilyen dokumentumot, ha azt a párt igazgatója, főtitkára választmányi elnöke, központi bizottsági vezetője és frakciófönöke is ellátta kézjegyével. De az igazgató is csak akkor ír alá, ha a főtitkár már aláírt, és a választmányi elnök is csak akkor, ha a választmány két elnökhelyettese együttesen szignózta a dokumentumot előtte. Minél bonyolultabb a szervezet és minél kényesebb a kérdéses dokumentum tartalma, annál összetettebbek lehetnek az aláírásokkal szembeni követelmények. Én például, kíváncsi volnék az európai uniós ratifikációs okmányra ilyen szempontból is. Hagyományos dokumentum esetében ez persze kevesebb gondot jelent, de elektronikus dokumentumok esetében alapos átgondolást kíván, hiszen ez esetben az aláírások nem helyezhetők el csak úgy, tetszés szerint a papíron.

> Az elektronikus aláírás több egy dokumentumlenyomat kódolt értékénél, hiszen ki kell fejeznie az aláíró szándékát, és bizonyítani kell az aláírás érvényességét.

Akármik is az elvárások, azok a többszörös aláírások két alapvető típusába oszthatók. Az első a független aláírások kategóriája, ami párhuzamos aláírásokat jelent. Ebben az esetben az egyes aláírások egymásutániságának nincs jelentősége. A második kategória a beágyazott aláírásoké, ahol az aláírások sorrendisége is fontos.
A legtöbb hétköznapi alkalmazás még nem kezel együttes és többszörös aláírásokat, egyelőre annak is örülni kell, ha egyáltalán kezel valamilyen elektronikus aláírást. Ha azonban olyan feladatot kapnánk, ahol egy szervezet vagy egy tranzakciós környezet jelenlegi bonyolult aláírási módszereit kellene leképeznünk elektronikus formára, alaposan fel kell térképeznünk az elvárásokat, és lehet, fel kell készülnünk a folyton változó előírások kezelésére is.

Aláírásformátum

Az előző részben bemutatott aláírástípusok mellett feltétlen szót érdemelnek az aláírásformátumok is, hiszen a két téma szorosan összefügg.
A jelenleg elérhető aláíráskezelő alkalmazások különböző formátumokat használnak, melyek közül a PKCS #7-es az egyik legelterjedtebb. Az elektronikusan aláírt levelek szabványa, az S/MIME is erre épül. Ez a formátum csak a legegyszerűbb szolgáltatásokkal bír. Többszörös aláírásokra egyáltalán nem ad módot, és az aláírás ellenőrzésére szolgáló adatok közül is csak a tanúsítási lánc egyes tanúsítványainak aláírásba foglalására ad lehetőséget.

Jóval fejlettebb képességeket tudnak felmutatni az XML Signature és XML Advanced Electronic Signature formátumok. Ezek lehetőséget adnak többszörös aláírások kezelésére, valamint időbélyegző és érvényességi információk aláírásba foglalására is.

Aláíráskezelő alkalmazás készítése

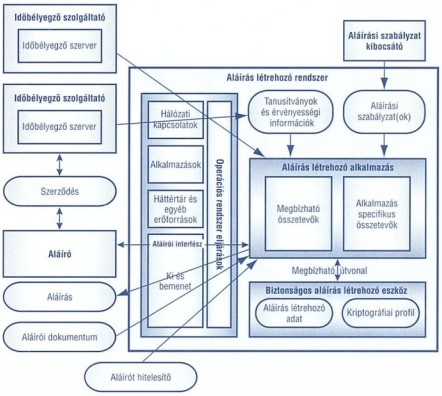
Eme igencsak hosszúra *(két és fél cikkre)* nyúlt búvártanfolyam után végre következik a lényeg, ami miatt a sorozatot elkezdtem. Elmerülünk az aláíráskezelő alkalmazások titkaiban. Először is emlékeztetőképpen veszünk pár példát arra vonatkozóan, mi mindenre kell gondolni, mielőtt nekiállnánk bármit is fejleszteni, majd átnézzük az aláíráslétrehozás és -ellenőrzés funkcionális modelljét, aztán - a következő részekben - az ezeket végző alkalmazások komponenseit, és műveleteit, az általuk kezelt adatokat, és az alkalmazásokkal szembeni biztonsági követelményeket.

Az aláíráskezelő alkalmazás tervezése

Az aláíráskezelő alkalmazás tervezésekor, illetve a tervezés megkezdése előtt a követelményjegyzék összeállításakor számtalan dolgot tisztázni kell. Az eddig elmondottakra visszautalva például:
El kell dönteni, hogy az alkalmazás hogyan kezeli az aláírási szabályzatot. Szabad formátumú szövegként definiáljuk azt; vagy gépileg feldolgozható nyelvként, esetleg vegyesen. Egy vagy több szabályzat szerint is működőképes lesz-e? Az előbbi esetben mi az a szabályzat, ami szerint működik, utóbbi esetben pedig, hogy milyen módon lehet hivatkozni az aláírási szabályzatra *(a lehetséges szabályzatok közül)* az aláírásban, illetve milyen módon választhatja azt ki az aláíró. El kell dönteni, hogy az ellenőrzés során hogyan történik a szabályzat feldolgozása, s milyen módon leírt szabályzatokat képes feldolgozni az alkalmazás. Látni kell, milyen aláírói szerepkörök és kötelezettségvállalások kezelésére kell felkészülni, szükség van-e többszörös aláírások kezelésére.
Tisztázni kell, hogy az alkalmazás milyen közegben fog működni: otthoni vagy irodai, netán nyilvános vagy mobil környezetben. Elosztott alkalmazás lesz-e *(az egyes komponensei hálózati kapcsolaton keresztül kommunikálnak egymással)* vagy minden komponense ugyanazon a gépen és közvetlen kapcsolódó egységeiben fog futni *(gép, kártyaolvasó stb.)*. El kell dönteni, hogy az alkalmazás intelligens kártyát, tokent vagy esetleg HSM-modult használ-e majd BALE-ként, vagy ezek közül több is előfordulhat, és ezek pontos típusait sem árt felmérni. Ha kártya, akkor milyen kártyaolvasó lesz: egyszerű, kijelzős, PINPAD-es, tudásalapú *(PIN/jelszó)* vagy biometrikus azonosítással? Tisztázni kell, hogy az alkalmazás teljesen automatikusan ellenőriz-e majd aláírásokat vagy szükség lesz humán interfészre. Az aláírás megtétele is történhet teljesen automatikusan, bár a jogszabályok szerint csak természetes személy hozhatja azt létre *(a természetes személy ugyanakkor valószínűleg felhatalmazhat ezzel a joggal egy alkalmazást)*. Ha lesz humán interfész, az egyetlen nyelvű lesz vagy netán többnyelvű. Tudni kell, hogy az aláírásellenőrzés folyamatát részben vagy teljesen átveheti-e egy harmadik fél *(tipikusan érvényességi szolgáltató)*. Látni kell, hogy milyen algoritmusok használata lehet szükséges az aláírás megtétele és ellenőrzése során, beleértve az összes lépést *(aláírás, tanúsítvány, visszavonási lista stb. ellenőrzése)*, s azt is, hogy mik ezeknek az algoritmusoknak a paraméterei. Nem mindegy például, hogy a szolgáltatói tanúsítványok 1024 vagy 2048 bites RSA-kulcsokkal vannak aláírva, esetleg DSA-algoritmussal, hiszen nem biztos, hogy a rendelkezésünkre álló fejlesztőeszközben ezek az eljárások implementálva vannak. Nem is beszélve az esetleges licenckérdésekről. Látni kell azt is, hogy milyen szolgáltatók és szolgáltatások igénybe vételére fog sor kerülni, s ezeknek mik a jellemzőik. Használva lesz-e időbélyegző, rendelkezésre áll-e azonnali érvényességi információs szolgáltatás, a tanúsítványok meddig vannak a tanúsítványtárban, s bár az aláíráshoz közvetlen nem kapcsolódik, a titkosítás követelményére is fel kell készülni. Ezek alapján kell meghatározni a szükséges aláírástípust *(lásd II. rész)* és az alkalmazott aláírásformátumot.

Az aláíráskezelés funkcionális modellje

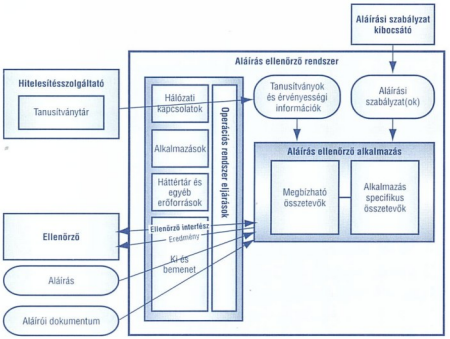
Innen kezdve az aláíráskezelő alkalmazásokat és eljárásokat nem egységesen vizsgáljuk, hanem két típusra osztva. Az aláírás létrehozása és ellenőrzése két eltérő folyamatot kíván, melyekkel szemben mások a biztonsági elvárások. Vannak ezek között természetesen átfedések, minthogy mindkét folyamat ugyanazon adatokat kezeli.



Aláíráslétrehozás funkcionális modellje

Az aláírás létrehozását egy aláírás létrehozó rendszer végzi, melynek két fő alkotóeleme az aláírás létrehozó alkalmazás és a biztonságos aláírás létrehozó eszköz. A kettő között egy megbízható útvonalnak kell léteznie a biztonságos adatcseréhez. A biztonságos aláírás létrehozó eszköz egy kriptográfiai profil szerint működik, s tárolja az aláírást létrehozó adatot. A rendszernek része mindaz a szoftver- és hardverelem, melyeket az alkalmazás és az eszköz használ, és amelyek „csak ott vannak". Így például az alkalmazást futtató gép és operációs rendszer maga, azok ki és bemeneti egységei, hálózati csatlakozásai, erőforrásai, és az azon futó egyéb alkalmazások. Ezek annak ellenére is fontosak, ha az aláírás létrehozásában nincs szerepük, mivel ez esetben is kockázati tényezőt jelentenek. Az operációs rendszer, valamint a gép ki és bementi egységei természetesen megkerülhetetlenek. Az aláírási szabályzatra már számos szót vesztegettünk: ez irányítja az alkalmazás működését. A hitelesítésszolgáltató szerepe itt másodlagos: ő biztosíthatja azokat a tanúsítványokat és érvényessé-

gi információkat, melyek az aláírásba kerülhetnek, az aláírás ellenőrzésének megkönnyítése végett. A hitelesítésszolgáltató és az aláíró között előfizetői szerződés áll fenn. Az időbélyegző szolgáltató az időbélyegzőt állítja el, mely az aláírás hiteles időpontját igazolhatja az aláírás részeként. Az aláíró egy aláírói interfészen keresztül kommunikál az aláírás létrehozó alkalmazással, mely interfész egyaránt tartalmaz bemeneti *(pl. billentyűzet)* és kimeneti *(pl. képernyő)* elemeket. A modell bemenete az aláírói dokumentum és az aláíráslétrehozó adat, kimenete pedig az aláírás maga. Az aláírót hitelesítő adatok közvetlenül, vagy az alkalmazás közvetítésével jutnak el a biztonságos aláírás létrehozó eszközhöz. Előbbire példa egy PIN-PAD-es kártyaolvasó, utóbbira pedig egy token. Az aláírót hitelesítő adat azonban nemcsak PIN-kód vagy jelszó lehet, hanem valami biológiai azonosító is.



Aláírásellenőrzés funkcionális modellje

Az aláírásellenőrzés modelljének fő eltérése a létrehozás modelljéhez képest, hogy az aláírásellenőrző rendszer nem tartalmaz biztonságos aláíráslétrehozó eszközt. Az aláírásellenőrző alkalmazás a hitelesítési szolgáltatóhoz abból a célból kapcsolódik, hogy az aláírás ellenőrzéséhez szükséges adatokat beszerezze *(amennyiben azok nem lettek az aláírásba foglalva)*. Az ellenőrzőhöz a kapcsolatot egy ellenőrzői interfész biztosítja, ami az aláírói interfész megfelelője. A modell bemenete az aláírói dokumentum és az aláírás, kimenete pedig egy eredmény, mely számos értéket felvehet. Lehet rendben, vagy lehet hibás, mely utóbbi értéket még alaposan tovább lehet cizellálni, mint az látni fogjuk.

Almási János
Almasi-Janos@dbrt.hu

Terminológia

Aláíró (fél)
Az a természetes személy, akihez a hitelesítésszolgáltató által közzétett aláírás-ellenőrző adatok jegyzéke szerint az aláírás-ellenőrző adat kapcsolódik.

Ellenőrző (fél)
Az a személy vagy automatizmus, aki vagy amely az aláírásellenőrző adat felhasználásával meggyőződik az elektronikus dokumentum hitelességéről és sértetlenségéről, valamint az aláíró személyéről.

Aláíráslétrehozó adat
Az aláíróra jellemző egyedi és egyéni adat, mely az aláírás megtételéhez szükséges. Tipikusan egy magánkulcs, mely egy tanúsítványhoz s az abban tárolt nyilvános kulcshoz *(aláírás létrehozó adathoz)* kapcsolódik logikailag.

Biztonságos aláíráslétrehozó eszköz - BALE
Olyan hardvereszköz, melynek segítségével az aláíró az aláíráslétrehozó adatok felhasználásával az elektronikus aláírást létrehozza. Az elektronikus aláírás létrehozásán kívül feladata az aláíráslétrehozó adat biztonságos tárolása is. Hétköznapi esetben egy intelligens kártya *(a kártyaolvasó nem tartozik az eszközhöz)*, különleges igények esetén egy hardveres biztonsági modul.

Hardveres biztonsági modul – HSM
Olyan professzionális aláíráslétrehozó eszköz, mely az általa nyújtott biztonság vagy aláírási kapacitás tekintetében nyújt többet egy egyszerű intelligens kártyánál vagy tokennél. Tipikusan egy, a gép rendszerbuszára csatlakozó kártya vagy egy, a géphez dedikált vonalon, esetleg hálózati kapcsolaton keresztül csatlakozó különálló egység.

Aláírót hitelesítő adat – aktiválási adat
Az aláíróra jellemző adat, mely a biztonságos aláírást létrehozó eszköz használatához szükséges.

Kriptográfiai profil
Elektronikus aláírás létrehozására és ellenőrzésére szolgáló kriptográfiai algoritmusok és egyéb releváns eljárások *(mint például hash és feltöltő függvények)* készlete, azok érvényes paramétereivel és beállításaival.

Tartalomformátum
Az aláírói dokumentum típusa, melynek lehetséges értékeire vonatkozóan az aláírási szabályzat megkötésekkel élhet. Elviekben bármilyen állománytípus lehet nem csak *(szöveges)* dokumentumra vonatkozó, de például végrehajtható szoftverkód vagy akár mp3 is. Ajánlott aktív kódot nem tartalmazó állománytípusokra szűkíteni az elfogadott értékeit *(pl. rtf, pdf, xml, txt, jpg)*.

Elliptikus görbe kriptorendszerek I. rész

Elliptic Curve Cryptography – elméleti háttér

Egy jó ideje egyre több helyen találkozhatunk az ECC betűhármassal, mint egy kriptorendszer megjelölésével. Egyre több előnyéről hallunk: az RSA-nál rövidebb kulcsokkal érhető el ugyanakkora biztonság, gyorsabb a működése, kisebb a memóriaigénye. Mindezek a SmartCard technológia biztonsági eszközeit is az ECC felé fordítják. De mi is ez? Elliptic Curve Cryptosystem, bár gondolom ettől most sokan nem lettek okosabbak, nekik *(is)* szól az a kétrészes cikksorozat, melynek első része következik.

Az elliptikus görbékről egyre gazdagabb irodalommal és egyre több ismerettel bírunk. A Certicom szerint az elliptikus görbéket, mint algebrai és geometriai elemeket az elmúlt 150 évben behatóan tanulmányozták és ezeken a tanulmányokon alapul a gazdag és mély ismeretek tárháza. Tekintve, hogy a Certicom Corporation **[certicom]** az ECC egyik vezéregyénisége, sajnos ez a megállapítás csak féligazság, mert az igaz ugyan, hogy az elliptikus görbék régóta ismertek, azonban kriptográfiai szempontból csak 15 éve vizsgálják őket. Az elliptikus görbék kriptográfiai alkalmazását – egymástól függetlenül – két kutató is javasolta: először Neal Koblitz *(University of Washington)* 1985-ben, majd tőle függetlenül Victor Miller *(IBM)*. A jelenleg használt PKI rendszerek *(szinte)* mindegyike a következő három probléma valamelyikének megoldási nehézségén alapul:

- A faktorizálás problémája – IFP *(~1970-től, integer factorization problem)*
- diszkrét logaritmus – DLP *(~1970-től, discrete logarithm problem)*
- diszkrét logaritmus az elliptikus görbék felett – ECDLP *(~1985-től, elliptic curve discrete logarithm problem)*

Sajnos jelenleg senki sem tudja biztosan, hogy e három probléma elég erős-e, de ennek ellenkezőjét sem bizonyította még senki. Jelenleg csak olyan megoldó-algoritmusokról beszélhetünk, amelyek „napjaink legjobb algoritmusai", de nem biztos, hogy elvileg is a legjobbak. Mindenesetre – figyelembe véve a ma ismert támadási módokat – az IFP megoldására ma ismert legjobb algoritmus messze lekörözi a ma ismert legjobb ECDLP megoldó algoritmust.

Az alábbi táblázatban három kriptorendszer általánosan vagy szabvány szerint használt kulcsméreteit láthatjuk. Az egy sorban lévő kulcsméretek közel azonos biztonságot nyújtanak *(a NIST ajánlása szerint)*. Látható, hogy a két legismertebb nyíltkulcsos algoritmus *(RSA és ECC)* kulcsméretei – azonos biztonság mellett – „köszönőviszonyban" sincsenek egymással...

A NIST javaslata az egyes kriptorendszerek kulcshosszának összehasonlítására:

ECC modulus	AES	RSAmodulus	RSA:ECC
112	56	512	5:1
161	80	1024	6:1
256	128	3072	12:1
384	192	7680	20:1
512	256	15630	30:1

Az a tény, hogy az ECC sokkal kisebb kulcsmérettel is biztonságos, a következőket vonja maga után:

- gyorsabb algoritmusok *(Azonban az RSA-aláírás ellenőrzése és az üzenet titkosítása szinte verhetetlen, ha kis nyilvános exponenst használunk, például e=65537!)*
- kevesebb továbbítandó és kezelendő adat
- kisebb tárigény
- kisebb tanúsítványok.

Mindezen vélt vagy valós előnyök miatt nem csak a kutatók foglalkoznak az elliptikus görbékkel, hanem az üzleti élet számára szolgáltatásokat nyújtó cégek is. Ez eggyel több ok, hogy mi is megismerjük legalább az alapvető fogalmakat és módszereket.

A következő alfejezetekben az elliptikus görbéket először a valós számok halmazán definiáljuk és megnézzük az értelmezett műveleteket, a műveletek származtatását, esetenként geometriai jelentését is. Ne ijedjünk meg a sok ábrától és képlettől, jóval egyszerűbb, mint amilyennek elsőre tűnik! *(Ettől függetlenül sajnos igaz, hogy az ECC matematikailag jóval bonyolultabb, mint az RSA vagy a Diffie-Hellman, ezért sokkal nehezebb megérteni és elmagyarázni vagy bizonyítani, igazolni a felhasználók felé... Ha valaki a cikk végére ér, és úgy érzi, hogy egy kukkot sem értett meg belőle, nyugodtan lapozzon vissza és kezdje elölről!)*

Valós számok halmazán járva – a görbe

Jelöljünk ki a koordinátasíkon egy **P***(x,y)* pontot! Ha a koordináták kielégítik az alábbi egyenletet, akkor a **P***(x,y)* pont az elliptikus görbe egy pontja:

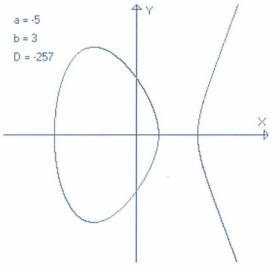
$$y^2 = x^3 + ax + b$$

Az elliptikus görbéknek még egy – definíció szerinti – pontjuk van: a végtelenben lévő pont, melyet „**O**" betűvel jelölünk. Nem keverendő össze a valós számok végtelenjével, ami megszámlálhatatlanul sokat jelent, míg az „**O**" pont inkább mérhetetlenül messze van *(nem mindig…)*. És ezekkel a pontokkal fogunk dolgozni! A velük végzett műveletek adják az elliptikus görbéken alapuló kriptorendszerek elemi műveleteinek nagy részét.

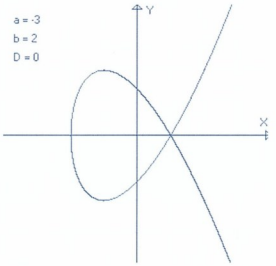
Az **a** és **b** paraméterek változtatásával újabb és újabb görbéket definiálhatunk, de nem mindegyik felel meg nekünk. Vannak olyan ***(a,b)*** paraméterpárosok, amelyekre igaz, hogy:

$$D = 4a^3 + 27b^2 = 0$$

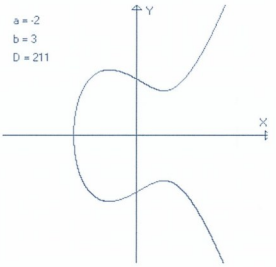
E görbék vagy elmetszik magukat vagy csúcsban végződnek. Most nem használjuk ezeket, nem tekintjük őket elliptikus görbének *(szinguláris görbék)*. Az alábbi három ábrán három görbét láthatunk azokra az esetekre, amikor D<0 , D=0 és D>0.



D<0, jó lesz!



D=0, ez bizony nem jó!



D>0, Ez is rendben van!

Azt állítottam, hogy a fenti görbék pontjaival fogunk dolgozni, lássuk hát miként?

Műveletek – geometriai megközelítésben

A görbék pontjaival mindössze három elvégezhető műveletet fogunk definiálni, lássuk most őket egyesével, részletesen!

Előjelváltás – ellentett képzése

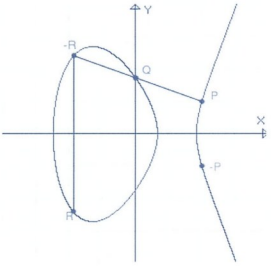
Egy ***P(x,y)*** pont ellentett párja **R = –P** nem más, mint a pont x tengelyre tükrözött képe, amely szintén rajta lesz a görbén: ***R(x,-y)***

Összeadás

Legyen a görbe két különböző pontja **P** és **Q**! E két pont összegét jelölje **R**, vagyis **R=P+Q**. A műveletet a következőképpen kell elvégezni:

1. Kössük össze a P és Q pontot egy egyenessel!
2. Az egyenes egy harmadik pontban metszi a görbét, ez a pont lesz **–R**.
3. E pont x tengelyre tükrözött képe az előjelváltás szabálya szerint szintén rajta lesz a görbén és ez lesz az eredmény **R** pont.

A művelet lépései az alábbi ábrán követhetők:



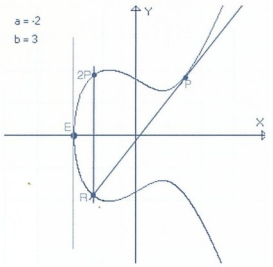
R=P+Q

P + *(–P)* = ?

Ha eddig egyszerűnek tűnik, akkor most próbáljuk meg **P** és **–P** pontot összeadni! Mivel **P** és **–P** egymás tükörképei az x tengelyre nézve, a rajtuk keresztül húzott egyenes párhuzamos az y tengellyel és sajnos nincs olyan harmadik pont, amiben metszené az elliptikus görbét. Az iménti ábrán látható még egy ilyen pontpáros *(**–R** és **R**)*: az őket összekötő egyenes szintén párhuzamos az y tengellyel és hiába hosszabbítanánk meg bármelyik irányba, nem fogja harmadik pontban metszeni a görbét. Majd csak a végtelenben… Ezért **P + *(–P)*** nem számítható ki a fenti módszerrel, de szerencsére van nekünk egy **O** pontunk, amely a végtelenben van. Definíció szerint **P + *(–P)* = O**, amiből az is következik, hogy elliptikus görbén értelmezve **P + O = P**. Hasonló azonosság a valós számok körében is van: x * 1 = x.

2P = ?

Egy pontot az előzőek mintájára adhatunk össze saját magával. Ha **P** és **Q** pontokat végtelenül közel visszük egymáshoz, akkor **P=Q** lesz. A **P** és **Q** ponton keresztül húzott egyenes pedig a **P** pontba húzott érintővé válik. A folytatást pedig ismerjük: az érintő metszi valahol a görbét és a metszéspont tükörképe lesz a **P** pont kétszerese.



Egy pont megduplázása

Sajnos azonban itt is van kivétel *(lásd a fenti ábrán **E** pontot)*: ha a pont az x tengelyen szíveskedik tartózkodni, akkor az érintő függőleges és nem metszi másik pontban a görbét. Ebben az esetben definíció szerint a pont kétszerese a végtelenben van, vagyis 2**E=O**. Ezek a pontok – melyeknek y koordinátája zérus – elég furcsán viselkednek, ha 2**E** után kiszámoljuk 3**E**, 4**E**, 5**E** stb. pontokat:

- 2**E** = O
- 3**E** = **E**+2**E** = **E**+**O** = **E**
- 4**E** = **E**+3**E** = **E**+**E** = **O**
- 5**E** = **E**+4**E** = **E**+**O** = **E** és így tovább.

Műveletek – algebrai megközelítésben

Az előzőekben az elliptikus görbén értelmezett műveleteket geometriai eszközökkel mutattuk be, de ez nem túl gyakorlatias a digitális számítógépek számára. Kicsit nehézkes érintőket és húrokat húzkodni, x tengelyre tükrözni, de szerencsére a geometriai műveletek eredményét ki is tudjuk számolni...

Előjelváltás – ellentett képzése

Ebben semmi újdonság sincs, ha a pont **P**(x,y), akkor **R** = **–P**:

$$x_R = x_P$$
$$y_R = -y_P$$

Összeadás

A levezetést mellőzve, csak a végeredményre hivatkozva: ha **P**(x_P,y_P) és **Q**(x_Q,y_Q) nem egymás ellentettje, akkor **R** = **P** + **Q**:

$$s = (y_P - y_Q) / (x_P - x_Q)$$
$$x_R = s^2 - x_P - x_Q$$
$$y_R = s(x_P - x_R) - y_P$$

P + (–P) = ?

Korábban láttuk, hogy ha a két összeadandó pont egymás ellentettje, akkor a rajtuk keresztül húzott egyenes egy függőleges egyenes, ami O pontban „metszi" a görbét. Az egyenes függőleges volták jól jelzi az is, hogy az s kiszámításához használt tört nevezőjében zérus van. *(Figyeljük meg, hogy s a PQ egyenes meredeksége!)*

2P = ?

Ha y_P nem zérus, vagyis a duplázandó pont nem az x tengelyen van *(lásd előző ábra **E** pontját!)*, akkor **R** = 2**P**:

$$s = (3x_P^2 + a) / (2y_P)$$
$$x_R = s^2 - 2x_P$$
$$y_R = s(x_P - x_R) - yP$$

A moduláris aritmetika közbelép – a görbe

A már megismert egyenletünket egészítsük ki egy kicsit! Ne a valós számokon értelmezzük, hanem a moduláris aritmetika szabályai szerint:

$$y^2 \equiv x^3+ax+b \pmod p, \text{ ahol } p \text{ prím}$$

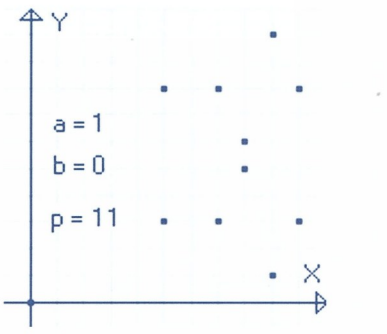
Vagyis ha az egyenlet mindkét oldala ugyanazt a maradékot adja p-vel történő osztás után, a **P**(x,y) pont a görbe pontjai közé tartozik. *(Igazság szerint a p modulus nem feltétlenül prím, bizonyos feltételek mellett akár összetett szám is lehet, de ez olyan speciális esetnek számít, amelynek megoldása sokkal egyszerűbb, mintha p prím lenne. Az ANSI X9.63 szabvány ezért egyenesen kizárja az összetett p modulussal definiált görbéket a kriptográfiai alkalmazásokból.)* Két további feltétel:

1. a **P** pont mindkét koordinátája kisebb legyen, mint p-1.
2. valamint $4a^3 + 27b^2 \bmod p <> 0$.

Néhány gyakorlati indok az áttérés mellet:

- A valós számokkal való számolás lassú és pontatlan. A moduláris aritmetika gyors és pontos, csak egész számokkal dolgozik.
- A „valós" görbének végtelen sok pontja van, a modulárisnak jóval kevesebb.
- A moduláris aritmetikában behatárolható a számok értelmezési tartománya, mert a műveletek operandusa*(i)* és eredménye mindig $0 \leqq x \leqq p-1$ közé esik.
- A moduláris aritmetika alkalmazása megnöveli a megoldások számát.

Ha a ponthalmazt az xy-síkon ábrázoljuk, már nem lesz olyan „szép", mint eddig, de megfigyelhető például, hogy továbbra is szimmetrikus. Igaz, most már nem az x tengelyre. A következő ábra p=11, a=1 és b=0 paraméterek által kijelölt „görbét" mutatja:



Íme az E: $y^2 \equiv x^3+1x+0 \pmod{11}$ „görbe"

Figyeljük meg, hogy:

- 11 darab pontja van a görbének,
- ebből egy darab az origóban *(mert b=0)*,
- 10 darab viszonylag véletlenszerűen, de az Y=5,5-re szimmetrikusan helyezkedik el, ezért
- minden X értékhez továbbra is kettő Y tartozik.

A görbe pontjainak száma most csak véletlenül egyenlő p-vel. Azokat a görbéket, amelyek pontjainak száma megegyezik p-vel, rendhagyó görbéknek *(anomalous curve)* nevezzük és gyakorlatilag az összes szabvány tiltja használatukat, mert létezik hatékony támadási módszer az ilyen görbét használó ECC-rendszer ellen. A pontok számát kiszámolni nem egyszerű feladat, de egy Hasse nevű úriember tétele szerint a pontok száma $(p+1) \pm 2\sqrt{p}$ között van. Egy pontra jellemző még egy szám, amelyet a pont rendjének hívunk: hányszor tudjuk összeadni saját magával, mielőtt **O**-ba érkezünk? Vagyis ha $n\mathbf{P}=\mathbf{O}$, akkor n a pont rendje. Ha a pont rendje megegyezik a görbe pontjainak számával, a pontot generátorpontnak hívjuk.

Műveletek a „görbe" pontjaival

Lássuk, hogyan kell értelmezni a már jól megismert műveleteinket moduloaritmetikával! Szerencsére nem sok újdonságot forgunk látni!

Előjelváltás – ellentett képzése

Ha visszaemlékszünk a valós számokon értelmezett görbére, ott egy **P***(x,y)* ellentettje az *(x,-y)* pont volt. Most sincs ez másként, csak modulárisan kell számolnunk: *(x, -y mod p)*, ami nem más, mint: *(x, p-y mod p)*. Ha megnézzük a fenti példa megoldásait, láthatjuk, hogy az egymással szemben lévő pontok y koordinátáinak összege mindig p=11.

```
Például (5,3) és (5,8) → 3+8 = 11
```

Tehát egy **R** = **–P** pont kiszámolása:

$$x_R = x_P$$
$$y_R = -y_P \bmod p.$$

Összeadás

Kicsit nehézkes most olyat értelmezni, hogy „kössünk össze" két pontot és keressük meg a harmadik metszéspontot, főleg hogyan „húzzunk érintőt"? Ezért a korábbi algebrai eredményeket vesszük át a moduláris aritmetika szabályai szerint:

$$s = (y_P - y_Q)/(x_P - x_Q) \bmod p =$$
$$\Rightarrow (y_P - y_Q)(x_P - x_Q)^{-1} \bmod p$$
$$x_R = s^2 - x_P - x_Q \bmod p$$
$$y_R = s(x_P - x_R) - y_P \bmod p$$

P + *(–P)* = ?

Ez továbbra sem változik: **P + *(–P)* = O**

2P = ?

Ha y_P nem zérus, vagyis a duplázandó pont nem az x tengelyen van, akkor R = 2P:

$$s = (3x_P^2 + a)/(2y_P) \bmod p =$$
$$\Rightarrow (3x_P^2 + a)(2y_P)^{-1} \bmod p$$
$$x_R = s^2 - 2x_P \bmod p$$
$$y_R = s(x_P - x_R) - y_P \bmod p$$

A probléma: diszkrét logaritmus

Minden kriptorendszer alapja egy olyan probléma, amit gyakorlatilag lehetetlen megoldani. Az ECC-nek is ilyen probléma adja a biztonságát, mégpedig a „diszkrét logaritmus elliptikus görbék felett" kiszámolásának problémája, röviden: ECDLP. *(1991-ben néhány kutató elkészítette az RSA algoritmus elliptikus görbéken alapuló változatát, de néhány évvel később többen is megmutatták, hogy az elliptikus RSA-nak [ECC-like RSA] nincs számottevő előnye a hagyományos RSA-val szemben. Az ECRSA problémája egyébként továbbra is a faktorizálás maradt.)*

Eddig lényegében két műveletet definiáltunk a görbén: pontok összeadása és egy pont duplázása. Egy pont sorozatos összeadásával ***(R, R+R, 2R+R, 3R+R)*** tulajdonképpen már szorozni is tudunk. Az így képzett **Q**=n**R** pontot a pont skalár szorzatának nevezzük, de n meghatározása a szorzat alapján nem egyszerű feladat főként, ha a görbét mod p felett értelmezzük.

```
Legyen egy elliptikus görbe egyenlete:
  y² = x³ + 9x + 17 mod 23
Mennyi az R=(16,5) alapú diszkrét logaritmusa
  Q = (4,5) pontnak? ( Q=nR )
Első megközelítésben n kiszámolásához addig
adogassuk össze a R pontot önmagával, amíg meg
nem kapjuk Q-t:
  1R=(16,5)   2R=(20,20) 3R=(14,14) 4R=(19,20)
  5R=(13,10) 6R=(7,3)    7R=(8,7)    8R=(12,17)
  9R=(4,5)
Mivel Q(4,5)=9R, ezért a Q pont R alapú diszkrét
logaritmusa mod 23 felett: 9.
```

Ha most valaki azt mondja, hogy a logaritmus a hatványozás inverze és nem a szorzásé *(főleg nem az összeadásé)*, akkor annak maximálisan igaza van. De...

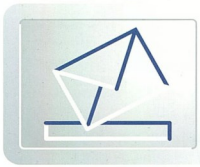
1. A racionális számokon értelmezett „hagyományos" logaritmus pontosan ugyanarról szól, mint például a Diffie-Hellman kulcscsere megoldása. Ha egy g számot x alkalommal összeszorzunk önmagával $(a=g^x)$, akkor az eredményből és g ismeretében: $x=\log_g a$ mod p , ha létezik. Ez a DLP.
2. Az ECDLP esetén adott P és Q pont a mod p felett értelmezett elliptikus görbén. Feladat: megkeresni x-et úgy, hogy xP=Q legyen, ha létezik ilyen szám.

Úgy tűnik a két probléma jelentősen eltér egymástól, hiszen eleve más műveletekről szólnak, az elsőben sorozatos szorzás van, a másodikban sorozatos összeadás. Azonban nálam hozzáértőbb matematikusok azt mondják, hogy a DLP szorzása és a ECDLP összeadása absztrakt módon ugyanaz. Általánosságban nem is szorzásnak meg összeadásnak hívják a használt műveleteket, hanem egyszerűen csoportműveleteknek *(group operations)*. Egy mérnöki szemléletű embernek ez hajmeresztőnek tűnik, ezért a „békesség kedvéért" ne keressünk most logikát az elnevezésekben, hanem vegyük tudomásul: az elliptikus görbéken értelmezett sorozatos összeadások *(=szorzás)* inverzét logaritmusnak hívjuk. Pont.

S mindezt mire lehet használni? Erről lesz szó a második részben!

(Folyt. köv.)

Virasztó Tamás
wacher@westel900.net
http://wacher.fpn.hu



SPAM, anti-SPAM

avagy Sajnos Pont A Mailboxomban gyülekező nemkívánatos adathalmaz

Napjaink egyik legkellemetlenebb problémájával foglalkozom ebben a cikkben, mégpedig a nem kért, de mégis tömegével kapott e-mail üzenetekkel, amelyek elárasztathatnak mindenkit, aki akár csak egyszer is küldött valaha egy e-mailt bárhova a saját hálózatán kívülre.

Elöljáróban vizsgáljuk meg, hogy miként is lehetséges ez, illetve a felhasználók által gyakran feltett kérdések egyikével élve: „Honnan tudják meg az én e-mail címemet?!". A kérdésben szereplő rejtett alany, sok esetben valóban rejtett marad, hiszen manapság már a SPAM-ek mögött közvetlenül nem a gonosz kis marketinges emberkét kell keresnünk, hanem egy nagyteljesítményű e-mail motort, amely többezres példányszámban generálja az üzeneteket. Olyan hatalmas mennyiségben születik ma már a reklám-SPAM, hogy ha csak egy minimális százalékukra érkezik is válasz, az már elegendő a nyereséghez. A vizsgálódást kezdjük a jó öreg postán, a régi – manuális – papírt és tollat igénylő levélfeladásnál. Tehát aki már adott fel a postán sima levelezőlapot, az tudja, hogy hogyan is néz ki az ilyesmi és mennyire „védett" a rajta található információ.

Ha a feladó neve és címe is szerepel, akkor valóban az összes adat a rendelkezésre áll, hogy legközelebb már egyéb célzattal is megkeressük a két felet. Kulcsszavakra keresve például a szövegből kiszűrhető egy adott témakör, amiben a feladó és a címzett érdekelt. Megvan minimum egy biztos cím is – a címzetté – és már mehet is a SPAM. A cikk során szeretném ezt az analógiát végig megtartani a postai és a számítógépes levelezés között, mert így könnyebben emészthetővé válik egy-egy sokszor bonyolultnak tűnő eset is.

Mert például az üzenet témájának védelmében a postán a kis levelünket egy borítékba bújtathatjuk az avatatlan tekintetek elől, míg ez a virtuális világban egy PGP-t, vagy valami hasonló „borítékot" jelent. A küldeménynek mindig lennie kell egy címzettnek, a feladó elhagyható *(sajnos)*, illetve ami még rosszabb: átírható, hamisítható. Az átvevő posta rajtahagyja a „kézjegyét" a levélen, mint ahogyan ezt teszik a levelezőszerverek is. Stb, stb. Egy nagy különbség van csak, mégpedig a levélküldés ára. A postán ugyanis levelenként célállomásfüggő díjat kell fizetni, amit viszont a mátrixban elhagytunk, vagyis nincs is definiálva ilyesmi a számítógépes világban. Ennélfogva máris megszülethet az első SPAM-ellenes megoldás: „Legyen fizetős az e-mail!"… Ám ha fizetni is kellene érte, sajnos az sem jelentene túl nagy korlátot. Jusson eszünkbe a kis szemetesdoboz a lépcsőházban, ami direkt ezért van odakészítve a postaládák közelébe. Pedig az abba kerülő küldeményeket nem ingyen hozta a postás. Képzeljük csak el, mi lenne, ha ingyen jöhetne az a sok-sok felesleges reklám, szórólap, figyelemfelkeltő színes kiadvány. Még a végén nem maradna hely a TechNET magazinnak…

Nos, gondolom ez sokakban elindított egy érdekes gondolatmenetet, ezért most ne is menjünk mélyebbre ebben, azon egyszerű okból kifolyólag, mert például a virtuális bélyeget igencsak nehéz lenne kezelni… Marad tehát a másik módszer, hogy megpróbáljuk kivédeni a SPAM-et. Ez nehezebb de egyben szebb feladat is, hiszen komoly kihívással kell szembenéznie annak, aki ilyesmire vállalkozik.

A SPAM

A rövidítés feloldására sok verzió látott már napvilágot. A teljesség igénye nélkül bemutatnék néhányat:

- Sending and Posting Advertising in Mass
- Stupid Pointless Annoying Messages
- Self Propelled Advertising Material

A szívemhez mégis az én verzióm – ami a cikk címében is olvasható – áll a legközelebb. Azért is, mert Sajnos Pont Az én Mailboxomban burjánzik a Föld nevű bolygó összes nemkívánatos e-mail üzenete. Van köztük általános reklám, van ami a pénztárcám tartalmára áhítozik a kongói kormány nevében, van ami több másodperc alatt jelenik csak meg, mert annyi kép és link van benne. De vannak olyanok is, amik igen kínos helyzetbe hozhatnak abban az esetben, hogyha egy nő kollégám áll a hátam mögött, amikor megnyitom őket. Ebből a fajtából van ám a legtöbb, napi 10, 20 vagy 50 is akár. Egyik „ízlésesebb" mint a másik.

Első gondolatra az átlagos felhasználó megnézi a feladó címét és kitiltja azt a mailboxából a Rules Wizard segítségével. Ez ideig-óráig jó megoldás lehet, de sajnos ugyanannak az üzenetnek a következő napon már más a feladója, hiszen mint tudjuk az átírható, módosítható. Így állandó csatározásban leszünk egy általunk nem is ismert láthatatlan ellenséggel, aki/ami folyamatosan megtalálja majd a postaládánkba vezető utat. Mert a mi kifelé irányuló leveleink mindegyike ott hirdeti magán nagybetűkkel a címünket, így ezen az úton mindig „visszatalálhat" hozzánk bárki. A probléma rendszergazdai szempontból is igen összetett, hiszen rendszerszinten sem érdemes a feladóalapú kitiltással vesződni. Ezen kívül komoly probléma lehet az is, hogy az ilyen levelek méretéből adódóan sok hely pazarlódik el a mailboxok tárolására szánt lemezen is.

Gondoljunk csak a nyári szünetben magárahagyott mailboxra, amire aztán rácsodálkozunk két hét múltán, hogy mi is történt, amíg távol voltunk. Van olyan is, amikor egy bizonyos méretkorlát elérése után nem fogad több levelet a postafiók, és természetesen pont a fontos levelek pattannak majd vissza a feladónak, miután betelt a rendelkezésre álló hely. Az alap SPAM-trükköket alkalmazó módszereket egy jól konfigurált mailszerverrel kivédhetjük. Ilyen például, amikor a feladó címe nem is e-mail cím, vagy a formátuma kissé rend- és szab-

ványhagyó. Vagy amikor a feladó mező nincs is kitöltve, illetve a hálózatunkon belüli másvalaki nevében küldöttnek álcázza magát egy üzenet, stb. De ha van rendes, szabványos feladó, címzett és vírusmentes tartalom, ebben az esetben a levél meg fog érkezni, mert látszólag semmilyen probléma nincs vele.

Megoldások, technikák

A SPAM-szűrés tehát valóban nehéz feladat, mert csak egy humán bázisú ítélőszék képes érdemben elbírálni, hogy egy adott levél SPAM-e vagy sem. Ezt a feladatot nem bízhatjuk számítógépre, legalábbis addig amíg a mesterséges intelligencia el nem éri ezt a szintet. Van egy módszer azonban, ami egészen jól használható, ez pedig a tartalomszűrés. Egy-egy adott stringre keres a program, és ha ilyet talál, beállítástól függően valamit tesz vele. Ez a megoldás bizonytalan, mert nem vizsgálja *(nem tudja vizsgálni)* a szövegkörnyezetet, amiben a kérdéses szekvencia szerepel, hanem csak szolgai módon figyeli annak előfordulását. Ebben az esetben el se mesélhetem e-mailben a kispajtásomnak, hogy milyen gonosz tartalmú levelet kaptam, mert azt is kiszűri majd, hiszen tartalmazza a vizsgált kifejezéseket. De például a rendszergazda sem küldheti át e-mailben az erről szóló jelentését a főnökének, mert ugyanúgy meg fog felelni a küldeménye a szűrési feltételeknek.

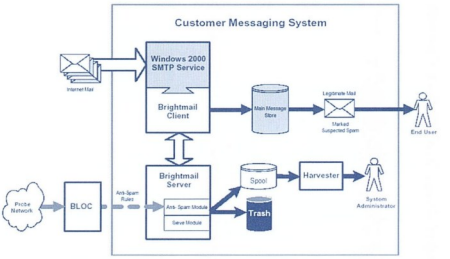
Ez a módszer tehát hatékony, de problémás és még plusz munkát is igényel az adminisztrátor részéről, mert folyamatosan nézegetnie kell a kiszűrt leveleket, hogy nincs-e közöttük valami tényleg fontos. A másik – és jelenleg a legfejlettebb – módszer, ha olyan szűrést alkalmazunk, ahol egy emberekből álló csoport előzőleg manuálisan eldöntötte, hogy egy-egy SPAM-nek látszó üzenet valóban az-e, vagy sem. Az így kapott eredményt egy adatbázisban közreadva, már csak azokat a jellemzőket kell keresnünk, amit ők előzőleg rosszként definiáltak, azaz benne vannak az adatbázisban.

Olyasmi ez, mint a víruskeresők által használt megoldás, amikhez mindig megjelenik szépen rendszeresen a frissítés. Mivel ez a módszer mindig csak válaszreakciót ad valamire, ennek természetéből adódóan előbb van a "kérdés" mint ahogy a válasz érkezik. Tehát kiindulásképpen először kapnia kell valakinek egy-egy ilyen üzenetet, hogy aztán eldönthessék róla, hogy bekerül a feketelista-adatbázisba vagy sem. Ez így még ezzel a „hátrányával" együtt is jóval hatékonyabb, mint az összes többi megoldás. Főleg az a változat, amit nekem volt szerencsém kipróbálni és azután élesben is üzembeállítani a hálózatunkon.

Ez a verzió ugyanis megoldja a SPAM fogadást és adatbázisba gyűjtést a gyártó cég saját hatáskörén belül, én már csak a végeredményt, a SPAM-mentes levelezést kapom. Hogy mindezt hogyan? Máris kiderül.

BrightMail AntiSPAM

Az **[Brightmail]**-es címen elérhető cég terméke megvalósítja az előzőekben ismertetett adatbázisalapú levélszűrést, és a Microsoft SMTP service-szel összebarátkozva lehetővé teszi, hogy a rossznak minősített levelek el se juthassanak a mailboxokba. A cég ugyanis egy saját kis hálózati szegmenst működtet ilyen célokra, ahol a létező összes SPAM-et megpróbálják begyűjteni. Feliratkoznak mindenféle listákra, gyanús weboldalakon adják meg e-mail címeiket, ezzel mintegy magukra vonják az összes mailbox-éhes SPAM-motor figyelmét. Ezt követően már csak el kell dönteni, hogy mi kerüljön be az adatbázisba. A telepített BrightMail Server, illetve a kliens ezt az információhalmazt veti össze az érkezett levelekkel. Ha bármelyik szerepel a feketelistán, a beállításoktól függően a következőket tehetjük: Forwardolhatjuk egy másik mailboxba, esetleg töröl-hetjük is, vagy file-ként gyűjthetjük egy könyvtárban őket.



A SPAM megfigyelése egy másik hálózatban történik

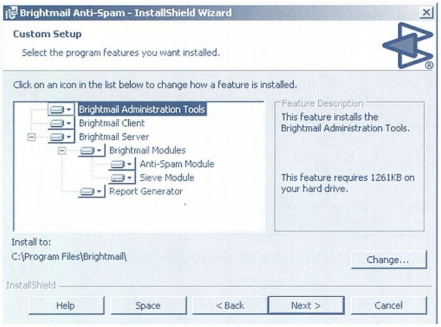
Van még egy lehetőség is a SPAM gyűjtésre, ez pedig az Exchange Foldering Agent. Ez egy olyan beépülő szolgáltatás, ahol minden egyes felhasználónak létrejön egy plusz folder a mailboxában – mondjuk SPAM néven – ahova gyűlik a nem fontosnak nyilvánított mailek sokasága. Így a felhasználó dönthet, hogy mit is kezd az így kapott üzenetekkel.

Ez persze egy sor kérdést felvet, miszerint van-e szükség egyáltalán az ilyen levelek tárolására? Kell-e a felhasználót terhelni ilyen problémával? Van-e a levelezőszervernek a SPAM-ek által fokozottabban kihasznált tárolókapacitása? Ezekre a kérdésekre én negatív választ adtam, így a Foldering Agent nem került telepítésre a mi hálózatunkban. Helyette egy elkülönített mailboxban tárolom a kiszűrt leveleket egy másik szerveren, ahol időnként megnézem, hogy mi az aktuális Top 10.

Mivel egy előre definiált adatbázis alapján dolgozik a filter, nagyon kicsi annak a valószínűsége, hogy egy valódi – nem SPAM – levél is fennakad a hálóján. Egy 250 felhasználós hálózaton másfél hét alatt 7000 elkapott SPAM üzenetből egy sem volt rendes levél. Tehát ez így 100%-os biztonságot jelent. Radikális csökkenés után persze, de volt azért egy-két SPAM ami mégis becsúszott. Hasonlóan mint egy új vírus, amikor nem friss a keresőadatbázis. De ebben az esetben jelentős pozitívumot mutat, hogy a napi 20-30 SPAM helyett heti 1-2 eshet csak be mailboxonként.

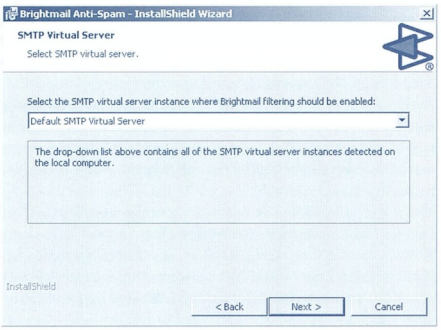
A BrightMail Telepítése

A szűrés tehát nem MX alapokon történik. Nem szükséges egy új mailszervert sem üzembe állítani, vagy elkonfigurálni a DNS MX bejegyzéseit. Ezzel szemben kell egy gép, amire a BrightMail Server Software kerül. Ez kérdezgeti majd folyamatosan a Spam Filter Rules adatbázist, amit a cég tesz elérhetővé a neten. A kliens pedig egy MS SMTP service-t futtató gépre kerül – például egy Exchange szerverre – amely majd kérdezgeti az előzőekben említett BrightMail szervert egy általunk is meghatározható IP porton keresztül. Lehet a Szerver és a Kliens egyszerre egy gépen is, ha elég erős a vas hozzá. A kliens installálásának viszont feltétele az SMTP service megléte az adott gépen. A szűrést az ilyen módon kicsit „kézbevett" SMTP service végzi és a beállításoknak megfelelően dönt az elkapott SPAM-ek sorsáról.



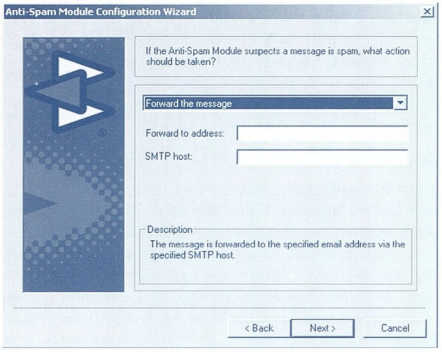
Az egyes összetevők szétválaszthatóak egymástól

A terhelés elosztásának szempontjából mindenképp érdemesebb a szervert és a klienst különvenni egymástól. A default kommunikáció kettejük között a 23456-os IP porton zajlik, illetve a Szerver komponens az Internet felől http-n keresztül jut hozzá a frissítésekhez. Ha van MS SMTP service a gépen, ahol az install MSI-t futtatjuk, a kliens install során ki kell majd választanunk a szűrni kívánt SMTP Virtual Server Instance-t, mert ugye ebből több is lehet egyszerre.



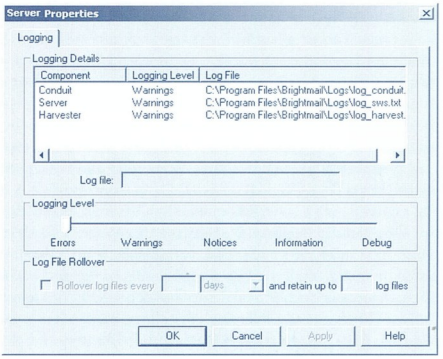
Választhatunk az SMTP Virtual Szerverek között

A szerver install opciói között látható egy Sieve Module névre hallgató program is, ami saját készítésű szabályok definiálására alkalmas arra az esetre, ha nem lennénk elégedettek a program beépített képességeivel. A mellékelt dokumentáció részletesen elmagyarázza, hogy miként és hogyan kell létrehozni egy-egy új szabályt a C nyelvre igencsak hasonlító eszközzel. Ritkán lesz rá szükség, viszont ha kell, ott van és nagyon hatékony.
Mint ahogy azt már említettem, az elkapott SPAM-ek sorsáról mi dönthetünk a szerver beállításainak megváltoztatásával. Én úgy gondoltam, hogy akár érdekességképpen is, de érdemes egy kicsit gyűjteni őket, hogy lássam, mi minden érkezett volna be ha nincs ez a software. Ezért egy teljesen más szerveren – ahol szintén van SMTP service, de nem vesz részt szervesen a hálózat levélforgalmában – egy külön mailboxban gyűjtöm a napi akár 1000 példánnyal is szaporodó e-mail áradatot.



Mit tegyen a Szerver ha SPAM-et észlel?

Egy megadott SMTP-n keresztül, egy általunk meghatározott mailboxba továbbítható a SPAM. Ennél a beállításnál vigyázzunk, hogy ne olyan címet adjunk meg, illetve olyan SMTP host-ot, ami mögött BrightMail féle ellenőrzés van, mert könnyen túlterhelhetjük egy ilyen lépéssel a szűrőmotort. *(Loop-ba kerülhet szegény.)*

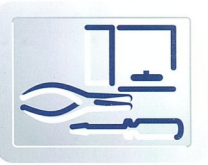


A logok beállításai is jól szabályozhatók

Az Administration Tools nevű komponens tartalmaz egy Performance Monitort, amivel az aktuális SPAM-szűrési számlálókat ellenőrizhetjük. Ezen kívül az Admin felületen győződhetünk meg arról is, hogy az adatbázis legfrissebb szabályai rendben megérkeztek-e hozzánk, illetve itt lehet még beállítani a logok részletességi szintjét is.

Füzesi Szabolcs
fuzesisz@osi.hu
MCSA

SQL Profiler



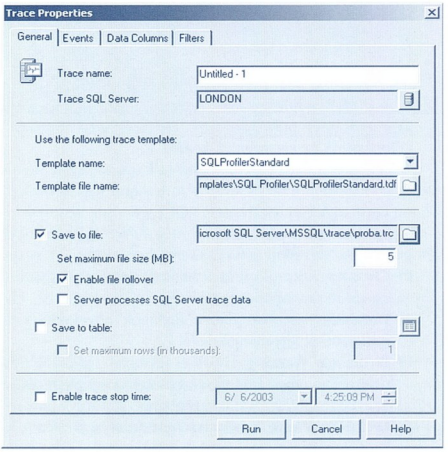
Aki SQL Serverrel foglakozik, akár fejlesztőként, akár rendszergazdaként időről-időre találkozhat „rejtélyes" esetekkel. Amikor valami okból nem úgy fut egy alkalmazásunk, ahogy mi elvárjuk, és sehol nem találjuk a hiba okát. Ilyenkor érdemes az SQL Profilerhez fordulnunk. Ez az eszköz az SQL Server részeként sokszor segítheti munkánkat.

Mióta SQL Serverrel dolgozom, többször fordultam bizalommal az SQL Profilerhez. Túlnyomó többségben hibakeresésre használtam, kihasználva a Profiler azon tulajdonságát, hogy az SQL Serveren történt minden eseményt megmutat, amire csak szükségem van. Bár ez közel sem minden, amit ezzel az eszközzel elérhetünk.
A Profiler elsődlegesen az SQL Server monitorozására szolgál. Az így kapott eredményhalmazt viszont már használhatjuk teljesítményelemzésre, hibakeresésre, tesztelésre vagy akár adott táblához indexvarázslásra is.
Az első lépésben indítsuk el a Profilert a start menüből vagy az Enterprise Manager Tools menüpontjából: egy üres ablakot kapunk. Indítsunk egy új szálat a New Trace ikonra kattintva vagy a File->New->New Trace menüpontot választva!
A megjelenő ablakon adjuk meg melyik SQL Serverhez szeretnénk kapcsolódni *(sőt egy pipa elhelyezésével azt is elintézhetjük, hogy az adott SQL Server el is induljon, ha esetleg le lenne állítva)*. A bejelentkezéshez pedig használjuk azt az autentikációs módot, amely számunkra a megfelelőbb.

Kapcsolódjunk a kiválasztott SQL Serverhez

Miután kiválasztottuk mondjuk a helyi szervert, állítsuk be a futtatás körülményeit. Az általános beállításoknál adjuk a Proba nevet a szálnak, és állítsuk be, hogy az eredményt mentse el a Profiler egy proba.trc fájlba. Ebből a fájlból bármikor visszaolvashatjuk az SQL Serveren adott időszakban lejátszódó eseményeket, elemezhetjük az így kapott adatokat, sőt vissza is játszhatjuk őket. Ez igen hasznos lehet akkor, ha egy adott rendszerben hirtelen megnövekszik a felhasználók száma, és szeretnénk megfigyelni mennyiben változott emiatt az SQL Server teljesítménye.

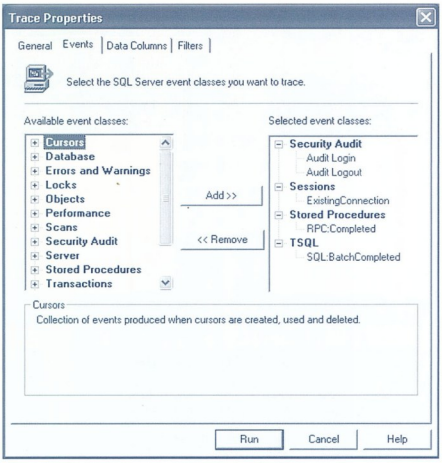
Persze nem muszáj a Profiler által visszaadott értékeket elmentenünk, ez csupán egy lehetőség.



Általános beállítások

Ha mégis úgy döntünk, hogy elmentjük az eredményhalmazt, fontos beállítani, maximum mekkora lehet az elmentett .trc fájl. Ha a Profilerben egy trace-t sokáig futtatunk, könnyen óriásivá duzzadhat az elmentett fájl. Ha az Enable file rollovert bejelöljük, a Profiler elkezdi felülírni a maximális méretét elért fájlt. Ha jobban szeretnénk SQL táblát használni az eredmények rögzítésére, erre is lehetőség van. Itt azt kell megadni, hogy melyik táblába mentünk és maximum hány ezer sor szerepelhet benne.
Ha egy trace-t csak elindítani szeretnénk, és azt akarjuk, hogy később automatikusan álljon le, az Enable trace stop time-nak adjunk meg egy másodpercre pontos értéket.

Az első lapon könnyedén túljutottunk, most állítsuk be milyen eseményekre vagyunk kíváncsiak!



Válogassunk az események közül

Alapértelmezésben a be és kilépések, létrejött kapcsolatok, a lefutott tárolt eljárások és Transact SQL parancsok kerülnek be a megfigyelt elemek közé. Persze még rengeteg mindenből válogathatunk. Nézzünk néhány eseményosztályt:

- **Cursors:** Cursorokkal kapcsolatos események. Például cursor létrehozása vagy törlése.
- **Database:** az adat- illetve logfájlok növekedését és csökkenését figyelhetjük vele.
- **Errors and Warning:** Hibák és kivételek figyelése. Például azt is megnézhetjük, hogy történik-e bejegyzés az ErrorLogban vagy az EventLogban.
- **Locks:** Zárolások. Igazán hasznos dolog a zárolások figyelése, főleg egy többfelhasználós rendszerben, ahol esetleg többen várnak arra, hogy valaki végre elengedjen egy táblát.
- **Security Audit:** ide tartozik minden, ami biztonsági hitelesítést igényel, vagy azzal kapcsolatos.
- **Stored Procedures:** A tárolt eljárásokkal kapcsolatos összes esemény. Például tárolt eljárás futtatásának kezdete, befejezése, újrafordítása.

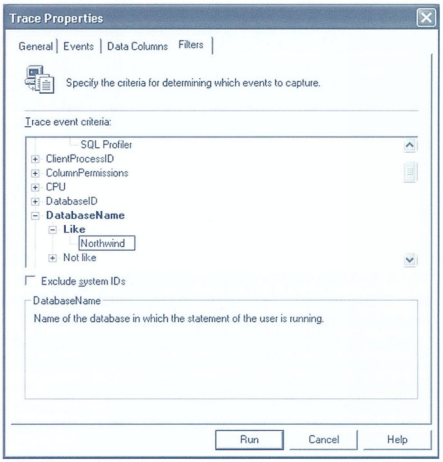
Egyelőre hagyjunk mindent alapbeállításon, majd később módosítunk rajta.

A Data Columns panelen megadhatjuk, milyen oszlopai legyenek a visszaadott táblázatnak. A legfontosabbak már a kiválasztottak között vannak, de hozzáadhatunk a listához újabb elemeket is a baloldali oszlopból.
Az ablakban az utolsó panel a Filter. Itt adhatjuk meg, hogy az összes SQL forgalom közül mi kerüljön a listánkba. Egy szűrő már be van állítva:

```
ApplicationName Not Like SQL Profiler
```

Azaz az SQL Profiler által generált eseményeket nem jelenítjük meg. *(Nem sok értelme lenne a vizsgált SQL utasítások közé bekeverni a vizsgáló eszköz tetteit. Ez olyan helyzetekhez vezethetne, mintha a látszerész arra a következtetésre jutna, hogy MI rosszul látunk, mert ő szemüveges.)*

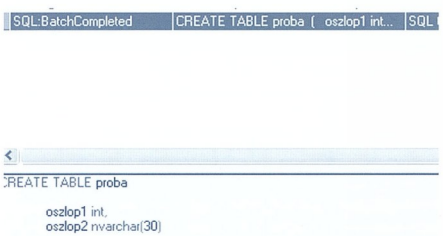
Ha további szűrőt szeretnénk megadni, megkeressük a megfelelő elemet *(például DatabaseName)* és a lenyíló ágban a Like vagy a Not Like feltételhez beírjuk a megfelelő értéket.



Szűrjünk a Northwind adatbázisra

Ha minden beállítással végeztünk, futtassuk a trace-t!
A legelső bejegyzés csak azt mutatja, hogy a trace elindult, majd a jelenleg létező összes kapcsolatot kiírja: ki jelentkezett be, milyen alkalmazásról és milyen beállításokkal. Ezután következnek a végrehajtott események. Itt csak azok az események jelennek meg, amelyeket előre beállítottunk. Emlékezzünk vissza, jelenleg ez az alapértelmezett, azaz a be és kilépések valamint az eljáráshívások és utasításvégrehajtások jelennek meg.
Nézzük, mi történik, ha létrehozunk egy táblát. Hajtsuk végre Query Analizerből a következő utasítást:

```
CREATE TABLE proba
(
        oszlop1 int,
        oszlop2 nvarchar(30)
)
```



Ekkor megjelenik egy sor a Profilerben, ahol láthatjuk, hogy végrehajtódott ez a parancs a Query Analyzerben, azonban hogy valóban létrejött-e a proba tábla, azt már nem látjuk. Mi

sem jobb példa erre hiányosságra, ha még egyszer végrehajtjuk ugyanezt az utasítást, a Query Analyzer kiabál, hogy nem tudja végrehajtani a parancsot, mert már létezik ilyen tábla, a profilerben viszont úgy jelenik meg, mintha minden rendben lenne.
Vegyünk fel tehát néhány eseményt a listába, hogy többet tudjunk, többet lássunk!
Először állítsuk le a trace-t *(eszköztáron a stop gomb vagy a File/Stop trace... menüpont)*, majd szerkesszük a tulajdonságokat *(eszköztáron a Properties gomb vagy a File/Properies menüpont)*. Vegyük fel az Errors and Warnings eseményosztályból az Exceptiont, valamint az Objects osztályból az Object:Created eseményt. Az oszlopokhoz adjuk hozzá az ObjectName és az Error oszlopokat, majd futtassuk újra a trace-t! *(Ekkor, ha az eredményhalmazt elmentjük, az adott fájl vagy tábla felülíródik.)*
Nézzük, mi történik, ha újra létre szeretnénk hozni a proba táblát anélkül, hogy töröltük volna előtte, vagy egy, az előzőhöz hasonló proba1 táblát készítünk.

Exception	2714	Error: 2714, Severity: 16, State: 6	SQL
SQL:BatchCompleted		CREATE TABLE proba (oszlop1 int...	SQL
Object:Created	proba1		SQL
SQL:BatchCompleted		CREATE TABLE proba1 (oszlop1 i...	SQL

Az Exeption is megjelenik, ha felvettük a megjelenítendő események közé

Rögtön láthatjuk, hogy a parancs futtatásával létre is jött a proba 1 tábla, viszont a proba tábla újbóli létrehozása 2714-es hibaüzenettel elszáll.
A fenti példából is látszik, hogy sok minden történhet a Serveren, és attól, hogy nem tudunk róla *(mert például nem megfelelő a hibakezelésünk)*, azért még ott van.

Visszajátszás

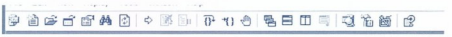
Ha már elmentettük egy adott időszak eseményeit, bármikor előszedhetjük, elemezhetjük, sőt vissza is játszhatjuk az adott eseményeket.
Készítsünk proba.trc fájlt a következő módon:

1. Nyitunk egy új trace-t a Profilerben.
2. Megadjuk, hogy az eredményhalmazt mentse egy proba.trc fájlba.
3. Az alapbeállításokkal kapcsolódjunk a helyi SQL Serverre.

Mivel én már az előzőekben készítettem egy proba táblát a Nothwind adatbázisban, ezt a táblát fogom használni. Query Analyzerből futtatva hajtsuk végre a következő utasításokat:

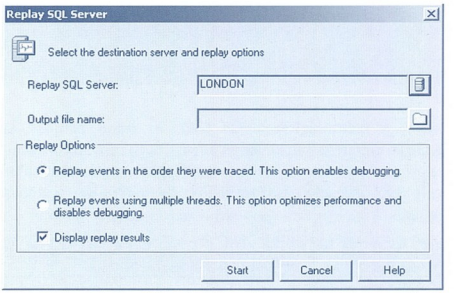
```
Select oszlop1 from proba
insert into proba select CategoryID as oszlop1, ''
as oszlop2 from Categories
delete from proba where oszlop1>6
Select oszlop1 from proba
Select oszlop1 from proba order by oszlop1
```

Majd állítsuk le a szálat. Ezek után töltsük vissza az így elmentett proba.trc fájlt *(File/Open menüpont)*.
Ekkor megváltozik az eszköztár, és a futtatás ikonjai helyett a visszajátszás *(replace)* ikonjai jelennek meg.



Fájlbetöltésekor megváltozik az eszköztár

Visszajátszhatjuk az összes lépést egyszerre, lépésenként, vagy csak a kijelölt sorokat. Amikor újra futtatjuk a tárolt lépéseket, meg kell adnunk, hogy mely szerveren szeretnénk ezt megtenni, hogy lehetőleg ne az éles adatbázisba tegyünk bele kétszer mindent. Továbbá megadhatjuk, hogy ugyanolyan sorrendben történjen a futtatás, mint az eredeti sorrend, vagy többszálú, párhuzamos futtatást szeretnék.



A visszajátszás beállításai

Gyakorlati hasznok

Most, hogy átfutottuk, mire is jó nagyjából a Profiler *(az Index Tuning Wizard kivételével, hisz erről már korábban olvashattak ezen oldalakon)*, nézzük a gyakorlatban mikor jöhet jól a használata.

Tesztelés

Mielőtt egy alkalmazást kieresztenénk a kezeink közül, kipróbáljuk, teszteljük, hogy tényleg megfelelően működik-e. Igaz, az általunk fejlesztett alkalmazásokban a hibák 20-30%-t valószínűleg nem találjuk meg, de azért a maradék 70%-ot érdemes kiszűrni.
Előfordult már velem is, és persze nem vagyok ezzel egyedül, hogy egy eljárás tökéletesen működött, amíg Query Analyzerben futtattam. Azonban amint egy webalkalmazásból hívtam meg ugyanezt az eljárást, már nem volt tökéletes. Nézzük meg egy konkrét példán keresztül *(a teljesség igénye nélkül)*, mi az, amibe ilyenkor belefuthatunk!
Készítsünk a Northwind adatbázisban egy GetProbaOszlop1 eljárást, ami visszaadja a proba tábla első oszlopából a feltételnek megfelelő sorokat.

```
CREATE Procedure GetProbaOszlop1
  @index int,
  @name varchar(30)
As
  Select oszlop1 from proba
  Where (oszlop2 like @name) and (oszlop1=@index)
GO
```

Ha ezt Query Analyzerben lefuttatjuk, mivel van hozzá jogunk, tökéletesen lefut. Azonban, mi van akkor, ha ugyanezt egy weboldalról szeretnénk megtenni.

```
...
Dim conn
Set conn = CreateObject("ADODB.Connection")
conn.ConnectionString =
"Provider=SQLOLEDB.1;Integrated
Security=SSPI;Initial Catalog=dbName;Data
```

```
Source=ServerName"

set cmd=server.createobject("ADODB.Command")
Set cmd.ActiveConnection = conn
cmd.CommandText = " GetProbaOszlop1 "
cmd.CommandType = adCmdStoredProc

Set param1 = Cmd.CreateParameter("@index",
adInteger, adParamInput, 4, request("index"))
Cmd.Parameters.Append param1

Set param2 = Cmd.CreateParameter("@name",
adVarChar, adParamInput, 30, request("name"))
Cmd.Parameters.Append param2

Set rs=cmd.Execute
...
```

Mivel a biztonsági beállítás SSPI, az adott parancs az IUSR_computername nevében fut*(na)*. Ha azonban nincs futtatási joga, akkor már kibukik a hiba. Ezt megfelelő hibakezeléssel könnyedén észrevehetjük, de a Profiler is szépen mutatja:

EventClass	Success	Permissions	TextData
SQL:BatchCompleted			exec GetProbaOszlop1

Ha futatthatnánk ezt az eljárást a Success oszlopban 1-es szerepelne

Adjunk futtatási jogot a IUSR_computername-nek a GetProbaOszlop1-re, és ezt a hibát ki is küszöböltük!
Most tegyük fel, hogy a fenti VBScript kódban szereplő index paramétert egy formból kapjuk. Mi van, ha ezt a paramétert a felhasználó nem töltötte ki? Akkor bizony default értékkel hívódik meg a GetProbaOszlop1. Tehát nem úgy, mintha NULL értéket tartalmazna, vagy egy üres stringgel lenne egyenlő, hanem default! Ilyenkor kliensoldalon egy hibaüzenetet kapunk, hogy hiányzik egy paraméterünk.

Audit Logout	1	
RPC:Completed	exec sp_reset_connection	
RPC:Completed	exec GetProbaOszlop1 default, 'a'	

exec GetProbaOszlop1 default, 'a'

Ha nem adunk a paraméternek értéket, akkor az alapértelemezett értékkel hívódik meg - ha van ilyen

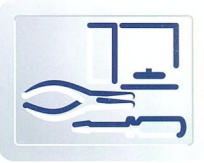
Bizony, mi nem állítottunk be alapértelmezett értéket erre a paraméterre, miért várnánk el, hogy visszaadjon bármit is ez a szegény eljárás? Ilyenkor vagy kliensoldalon dolgozzuk fel ezt az eshetőséget, vagy kicsit átszabjuk a tárolt eljárást.
Előfordulhat az is, hogy elvileg minden paraméter és a jogok is stimmelnek, de mégis elveszik valahol a recordset tartalma. Ilyenkor is jól jön a Profiler. Ha az SQL Serveren minden rendben, azt az eredményhalmazt kapjuk vissza, amit kell, akkor bizony valahol kliensoldalon van a hiba!
És folytathatnám a példák sorolását, amelyek alátámasztják a Profiler hasznosságát.

Karbantartás, teljesítményelemzés

Bármilyen szépen is működött egy felhasználó esetén az általunk készített alkalmazás, ez még nem jelenti azt, hogy 20-30 felhasználó esetén is ugyanolyan jól fog üzemelni. Sőt lehet, hogy már 2 júzer esetén is percekre „behal".
Ilyenkor érdemes elmenteni időnként az SQL Serveren történő eseményeket: a Profiler pont jó erre.
Aki villámolvasó, valószínűleg azzal is megbirkózik, ha realtime figyeli a becsapó kéréseket, zárolásokat, zárolás feloldásokat, hibaüzeneteket, parancsvégrehajtásokat. Valljuk be, azért a többségünk nem olvas 10-20 vagy több sort egyszerre, bár járnak közöttünk igazán profik, akik valószínűleg egy másik bolygóról érkeztek. Viszont az elmentett trace-t már mi, emberek is könnyedén elemezhetjük.
Jó tudni, hogy mely lekérdezések futottak legtöbbször. Kiderülhet akár az is, hogy az általunk készített index egy táblán nem éppen a legmegfelelőbb. Megtalálhatjuk azokat a parancsokat, amelyek a legtovább futnak, a legtöbb erőforrást igénylik, így célszerű optimalizálni őket, ha eddig nem tettük meg.
Mindezt pedig úgy, hogy – hála a szűrőknek – csak a számunkra szükséges sorok jelenjenek meg. Persze nem elég hangsúlyozni, hogy tudnunk kell, mit keresünk!

Borsi Katalin
bobo@netacademia.net

Objektum Orientált Alkalmazásfejlesztés I.



OOP, OOA, UML, DP, RUP, ER, ORM,... Sokáig folytathatnám még a HBR-eket ☺, amelyek a szoftvertervezés és -fejlesztés során felmerülnek, és amelyek tengerében nagyon könnyen el lehet veszni. Az útkeresés során látni fogjuk, hogy nem elég, ha az ember ismeri a rövidítések jelentését, a lexikális tudás mellett rengeteg egyéb képességre kell szert tennünk.

Az említett rövidítésekkel nap mint nap találkozhatunk munkánk során. Vannak, akik otthonosan mozognak ebben a világban, sokan pedig az első, gyors sikerélmény elmaradása miatt örökre kiábrándulnak az ide kapcsolódó módszerekből. Vannak azonban, akik szeretnék megtanulni, viszont nem tudják, hogyan induljanak el. Számukra szeretnék egy kis segítséget nyújtani.

A programozás kezdetben egészen mást jelentett, mint napjainkban. Informatikai léptékkel számolva ma már történelemnek nevezhető, amikor az operátorok teljhatalmúak voltak abban az értelemben, hogy minden egyes számítógép-hozzáférés rajtuk keresztül történt, és ha valami miatt nem fordult vagy hibás futási eredményt produkált programunk, kérhettünk új időpontot, és kezdődött elölről minden. A szoba- sőt csarnokméretű gépek mindemellett annyiszor meghibásodtak, hogy már önmagában ez is lényegesen rontotta a hatékonyságot. Ezt az időszakot már csak elbeszélésekből ismerem, de el tudom képzelni, milyen lehet ily módon fejleszteni például egy háromrétegű adatbázis-alkalmazást... (*tudom-tudom, sehogy!*)

A következő lépés az volt, hogy a fejlesztők már önmaguk is odaférhettek a számítógépekhez. Ez természetesen előrelépést jelentett, de a gépek továbbra is osztottak voltak, így a programozók még mindig csak meghatározott időszeleteket kaptak. Ezt az időt kellett úgy beosztani, hogy minél többet haladjon a fejlesztés, s ha az idő lejárt, lehetett újra feliratkozni – esetenként a következő hónapra. Abban az egyben biztos vagyok, hogy akkoriban munka közben igencsak kevés kávészünetet tartottak...

Egyszer aztán elérkezett az az időszak is, amikor a fejlesztők saját gépen dolgozhatnak. Annak idején jómagam Commodore-on kezdtem, ahol már az is művészet volt, ha az ember le akarta menteni a munkáját, vagy vissza akarta tölteni a kazettáról *(játszani azért már akkor is jókat lehetett)*. BASIC-ben próbáltunk nagyot alkotni, a sorszámozott programsorok és a GOTO-k között igyekeztünk eligazodni.

Aztán egyszer csak megismerkedtem a PC-vel és a Pascal nyelvvel. Azt hiszem, innen kezdve nem volt megállás...

Programnyelvek osztályozása

De vajon mi az, ami egyik programnyelvet megkülönbözteti a másiktól? Akkoriban óriási élmény volt, ha egy BASIC utasítássorozat működött, ma pedig már csak csendben mosolygok rajta. Vajon miért?

Az általam most vizsgált programnyelvek mind az iteratív nyelvek családjába tartoznak. A másik nagy csoport a deklaratív nyelvek csoportja, ide tartozik pl. a Prolog, az SML, az SQL vagy az XSL. Ezek közös tulajdonsága, hogy leíró nyelvek, azaz adott problémára vonatkozó statikus tényeket írnak le, semmit nem mondanak arról, hogy ezeket a statikus tudáselemeket milyen módon *(mely eljárás szerint)* kell a problémamegoldás során felhasználni. Többnyire a matematikai logika kifejezésmódjára támaszkodnak, így bizonyos kijelentések alapján automatikusan új kijelentések vezethetők le. Ez a lehetőség szolgál a mechanikus tételbizonyítás alapjául is. Fontos alkalmazási területként említhető a mesterséges intelligencia, a természetes nyelvek feldolgozása, elektronikus berendezések tervezése, ellenőrzése, analizálása, stb.

> Az objektumorientált paradigma segítségével gyorsabban és biztonságosabban érhetünk célt a szoftverfejlesztés területén.

Az iteratív nyelvek ezzel szemben a megoldás hogyanjára helyezik a hangsúlyt: nekünk kell megmondanunk, mit milyen algoritmus alapján kell végrehajtani.

Ez persze nem egyszerű feladat...

Kezdetben a programozók még nem ismerték az objektumorientált megközelítést: a programok függvényekből és metódusokból álltak, egyetlen óriási blokkba szerveződve. Később kialakult a strukturális megközelítés, azaz lehetőség volt arra, hogy a program különböző részeit különböző modulokba, különálló fájlokba helyezzük, növelve ezzel az áttekinthetőséget és a kezelhetőséget. Ez a megoldás már szebb és használhatóbb volt, azonban még mindig nem az igazi...

1968-ban a NATO szoftverfejlesztési konferenciáján vezették be a szoftverkrízis fogalmát. A fejlesztők számára világossá vált, hogy az egyre bonyolultabb programok már nem fejleszthetők a régi strukturált módszer segítségével, vagyis a minőséget nem lehet tartani. A „tünetek", amelyek alapján erre a kijelentésre jutottak, a következők voltak: a rendszereket egyre nagyobb késéssel szállították, működésük megbízhatatlan volt, és egyre kevésbé teljesítették a megrendelők igényeit. A válságból kivezető út már csak valamilyen új, forradalmi szemlélet, módszer megjelenése lehetett. Így született meg az objektumorientált paradigma, amelynek segítségével gyorsabban és biztonságosabban érhetünk célt a szoftverfejlesztés területén.

Természetesen nem valószínű, hogy ezzel egyszer s mindenkorra megoldódott a szoftverfejlesztők összes gondja. A szélе-

sebb lehetőségeknek köszönhetően a szoftverekkel támasztott igények egyre nőnek, és az aktuális technológia, a módszertanok nem biztos, hogy lépést tudnak tartani ezekkel. Ilyenkor újabb kivezető utat kell keresni...

Az objektumorientált módszertan

De mit is jelent valójában ez az új szemléletmód? Mi tulajdonképpen az objektumorientáltság?
Először is azt kell tisztáznunk, mit takar a moduláris programozás fogalma, s csak ezután érdemes továbblépni. A „szép" szoftver lényeges tulajdonsága, hogy áttekinthető, érthető, jól kezelhető modulokból álljon. Ehhez alapvetően két eszköz áll rendelkezésünkre: az absztrakció és a dekompozíció. Az absztrakció során a feladat szempontjából lényeges dolgokat kiemeljük, a lényegteleneket pedig elhagyjuk, a későbbiek során nem vesszük figyelembe őket. Például ha egy autót akarunk modellezni, lényeges a típusa, az ajtók száma, a légzsák megléte, de például ha a feladat csupán az autó biztonságának vizsgálata, a színe nem lényeges tulajdonság, ezzel nem kell foglalkoznunk. *(Hacsak nem pszichológiai tesztet végzünk, és azt akarjuk bebizonyítani, hogy a kicsi, piros női autók a legveszélyesebbek az állatvilágra...)*
A dekompozíció során az így redukált feladatot részfeladatokra, modulokra bontjuk, hiszen a kisebb részeket sokkal egyszerűbb áttekinteni és kezelni. A részfeladatok megoldása után azokat ismét egy egésszé összerakva kapjuk az eredeti feladat megoldását.
A moduláris programozásnak több alapelve is van, ezek közül a legfontosabbak:

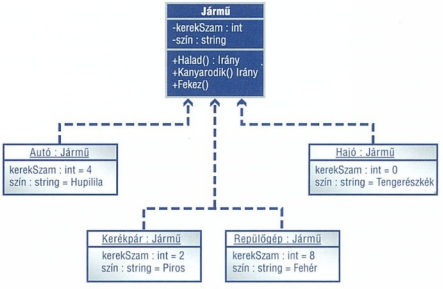
- „Oszd meg és uralkodj": A modulok egymás működésébe nem szólhatnak bele, de elvárható, hogy a saját feladatukat tökéletesen végezzék. A modulok belüli kötéseknek erőseknek *(kohézió)*, a modulok közöttieknek pedig lazáknak kell lenniük *(csatolás)*. Így a program sokkal áttekinthetőbb lesz, a hibák egyszerűbben kiszűrhetők és javíthatók.
- Az információ elrejtése: Az egyes modulok csak a saját adataikon dolgozzanak, csak akkor használjanak közös adatokat, ha az feltétlenül szükséges.
- A döntések elhalasztása: Csak akkor hozzuk meg a döntéseket, amikor már feltétlenül szükséges, és minden ismeretünk megvan hozzá. Máskülönben előfordulhat, hogy valamely későbbi szakaszban, az újabb ismeretek megszerzésével felül kell bírálni döntésünket.
- A döntések kimondása: A feladat megoldása során minden döntést dokumentáljunk. Ha ezt elmulasztjuk, fennáll a veszélye, hogy később nem emlékszünk, rosszul emlékszünk, esetleg kollégáinkkal kerülünk ellentmondásba.

A modulokra bontás kétféleképp történhet: ha fentről lefelé építkezünk, az eredeti feladatot bontjuk egyre finomabb és finomabb részekre, modulokra; ha alulról felfelé haladunk, a szoftvert már meglévő, kész modulokból állítjuk össze.

Az objektumorientált gondolkodásmód nagymértékben hasonlít az emberi gondolkodáshoz: a körülöttünk lévő tárgyakat, objektumokat csoportosítjuk, rendszerezzük, a közöttük lévő kapcsolatokat próbáljuk felderíteni. Az objektumorientált *(OO)* program kommunikáló objektumok összessége, ahol minden objektumnak jól meghatározott szerepe és felelőssége van. Az objektumban tárolhatunk adatokat, amelyek a feladat végrehajtásához szükségesek, illetve funkciókat, amelyek az adatokon végzett műveleteket definiálják. Az objektummal az interfészén keresztül kommunikálhatunk, továbbá az objektumunk mindig valamilyen állapotban van.
Példaként maradjunk az autós példánál. Egy autóobjektum adatai lehetnek az ajtók száma, a légzsákok száma és elhelyezkedése, a motortérfogat stb. A funkciók az indulás, fékezés, kanyarodás és így tovább. Az autó interfésze a kormány és az egyes pedálok, ezeken keresztül adjuk meg, mit és milyen paraméterekkel szeretnénk végrehajtatni *(például a Kormány interfészen keresztül meghívjuk a Kanyarodj metódust a [jobb, 90°] paraméterekkel)*. A kanyarodás során az autó állapota megváltozik, például az [Északra_Halad] állapotból átkerül a [Keletre_halad] állapotba.
Az alkalmazás tervezése során fő feladatunk tehát az objektumok és osztályok egyértelmű azonosítása, feladataik meghatározása, és kapcsolataik felderítése.

Objektumok és osztályok

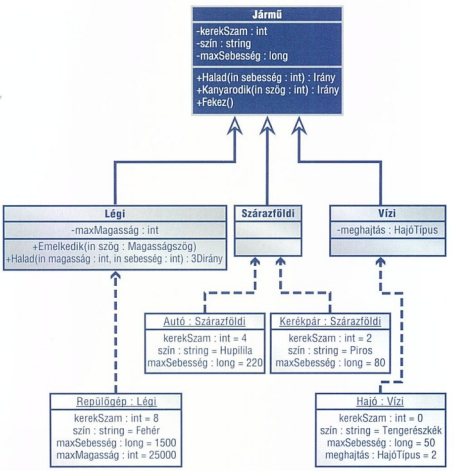
Az objektumokról már esett néhány szó, lássuk, mik is azok az osztályok, amelyek szintén elkerülhetetlenek az OO világban.
Képzeljünk el ismét egy autót. Természetes emberi módon gondolkodva lássuk, milyen osztályba, kategóriába sorolnánk az autót: az autó például egy jármű. Természetesen több járművet is ismerünk: kerékpár, repülőgép, hajó, stb. Ebben az esetben azt mondjuk, hogy a Jármű egy osztály, az autó, kerékpár stb. ennek a Járműosztálynak a példányai.



A Járműosztály és példányai

Mint az emberi gondolkodásban is, az OO programozásban is minden objektum jól meghatározott osztályba sorolható már a létrejöttekor.
Természetes módon adódik a kérdés: egy objektumot sorolhatunk-e egyszerre több osztályba is? Mondhatjuk-e például azt, hogy a kerékpár egyszerre jármű és sporteszköz? Ezáltal természetesen szeretnénk, ha mindkét osztály tulajdonságait, jellemzőit magában hordozhatná a programunkban is éppúgy, mint a valóságban.
Ez a megközelítés azonban igen sok problémát felvet: mi történjen akkor, ha a két osztálynak vannak azonos adatai vagy metódusai? Hogyan döntsük el, hogy mikor melyiket alkalmazzuk? A kerékpár honnan tudja, hogy őt most sportolásra vagy belvárosi közlekedésre használják?

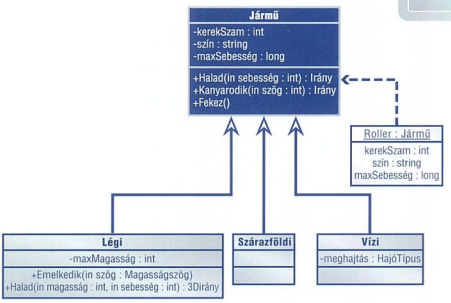
Ezen megfontolások alapján az objektumorientált programnyelvek többsége nem engedi a többszörös öröklődést, azaz minden objektumot egyetlen osztályba sorolhatunk, és minden osztály egyetlen másik osztályból származhat.
Származhat? Vajon mit jelent ez? Mint már említettem, nem elég az objektumok és osztályok azonosítása, ezek kapcsolatait is fel kell térképeznünk a tervezés folyamán. Az osztályok közötti kapcsolat egyik típusa az öröklődés, ami a következőt jelenti: Adott két osztály, nevezzük az egyiket A-nak, a másikat B-nek. Akkor mondjuk, hogy A ősе B-nek *(illetve B az A-ból származik)*, ha közöttük szülő-gyermek viszony áll fenn, azaz a gyermekosztály örökli a szülőosztály minden attribútumát és metódusát. Ezek a származtatott osztályban felülírhatók, és újabb tagokkal egészíthetők ki.
Például előfordulhat, hogy nem elég egy Járműosztályt definiálnunk, szükség van arra, hogy újabb osztályokat vezessünk be a vízi-, légi- és szárazföldi járművek megkülönböztetésére. Ekkor a struktúradiagram így nézhet ki:



A járművek statikus struktúradiagramja több járműosztály esetén

Jól látható, hogy például a Légi típusú Repülőgépnek továbbra is megvannak azok az attribútumai, amelyek a Járműosztályhoz tartoznak, de van egy újabb, maxMagasság attribútuma is, amely a légi járművek sajátossága. Hasonlóan definiálhatunk saját, az ősosztályhoz nem tartozó attribútumokat a többi osztály esetén is. Az is igaz továbbá, hogy a metódusok hasonlóan viselkednek: a fenti ábrán nem látható ugyan, de a Repülőgép a Halad, Kanyarodik és Fékez tevékenységeken kívül újabb, Emelkedik metódussal is rendelkezik.
A Légi osztályban ezen kívül felüldefiniáltuk a haladást: egy repülőgépnél nem elég a síkbeli irányt megmondani, mint például az autóknál, háromdimenziós adatra van szükség, hiszen emelkedhet illetve süllyedhet is. Ezért a Halad metódus nem Irány, hanem 3Dirány típusú lesz, és paraméterezése is másképp alakul, mint az ősosztályban.

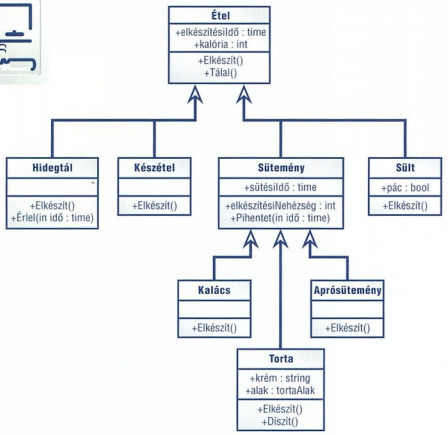
Fontos még bevezetni az absztrakt osztály fogalmát is. A fenti modellben elképzelhető olyan eset, amikor egy objektum közvetlenül a Jármű osztályból származik, s nem soroljuk egyetlen al-osztályba sem:



Közvetlenül az alaposztályból származó objektum (Roller)

Ez a megoldás helyenként szükséges és célravezető lehet, azonban sok esetben hátulütői is lehetnek: ha nem határozzuk meg közelebbről a Roller osztályát, és ezáltal típusát is, nem rendelkezünk konkrét, lényegre törő adatokkal és metódusokkal kezeléséhez. A Rollerről például csak annyit tudunk, hogy egy jármű, de nem tudjuk, hogy szárazföldi, vízi vagy légi-e, esetleg egy negyedik, eddig meg nem határozott osztályba sorolható *(pl. földalatti)*.
A másik tipikus igény az, hogy szeretnénk olyan osztályokat létrehozni, amelyekben definiáljuk, hogy leszármazottaiknak milyen attribútumokat és eljárásokat kell megvalósítaniuk, azonban ezeket nem feltétlenül konkretizáljuk az ősosztály szintjén, a metódusok törzse üres is maradhat. Ez olyankor lehet hasznos, ha például azt tudjuk, hogy minden Jármű halad valahogyan, de egészen biztosan tudjuk, hogy ezt másképp teszi egy repülőgép és másképp egy evezős csónak. Szeretnénk továbbá azt is, hogy ezen osztályon keresztül tudjunk hivatkozni az alosztályok példányaira is, azaz ha létrehozunk vízi és légi járműveket, azok elérhetők legyenek a Jármű osztályon keresztül is.
Ezekre a problémákra jelent megoldást az absztrakt osztály. Ezek az osztályok nem példányosíthatók, azaz ha a Járműosztály absztrakt, a fenti Rollerobjektum nem hozható létre közvetlenül belőle.

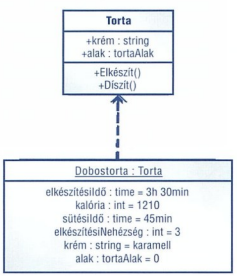
Lássunk egy példát erre is!
Vonatkoztassunk el most a járművektől, és evezzünk más területekre! Tegyük fel, hogy egy szakácskönyvet szeretnénk modellezni. Ebben a szakácskönyvben ételrecepteket tárolunk, a különféle koktéloktól, tablettás boroktól és paradicsomlevektől eltekintünk. Az ételek mindegyikére jellemző, hogy valamilyen módon készíteni kell, ehhez bizonyos időre van szükség, és van kalóriatartalma. Vannak azonban különböző típusai is az ételeknek: hidegtálak, készételek, sültek, sütemények, stb., és esetenként ezek az osztályok is tovább finomíthatók.
Ahhoz ragaszkodunk, hogy minden egyes étel a lehető legszűkebb osztályba tartozzon, így az Étel alaposztály absztrakt lesz: így egyrészt nem tudjuk példányosítani, másrészt rögzíthetjük benne a szükséges metódusokat. Lássuk tehát, hogyan néz ki a struktúradiagramunk:



Az ételreceptek statikus struktúradiagramja

Az Ételosztály tehát absztrakt, ezt jelöli nevének dőlt betűs írása. Ebből származnak a különböző ételosztályok, s a Sütemény osztálynak további leszármazottai vannak. Jól látható, hogy az Elkészít() metódus minden leszármazott osztályban implementálva van, ezen kívül, ahol szükséges, újabb metódusok jelennek meg.

Vegyünk fel egy objektumot, amely például egy Dobostortát modellez:

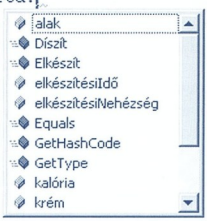


A Dobostorta objektum

Jól látható, hogy az objektum nemcsak a közvetlen ősének *(szülőjének)* attribútumaival rendelkezik, hanem az összes ős attribútumaival is. Ez ugyanígy igaz a metódusokra is: nemcsak a Torta osztály metódusai hívhatók meg rajta *(Díszít és Elkészít)*, hanem a Sütemény osztály Pihentetése és az Étel alaposztály Tálalása is.

Ha legeneráljuk a megfelelő kódokat, és létrehozunk egy dobostortát, amely a Torta osztály egy példánya, a következőt láthatjuk:

```
Torta dobosTorta = new Torta();
dobosTorta.
```



A dobostorta attribútumai és metódusai

Vagyis valóban láthatjuk az összes ősosztály valamennyi publikus tagváltozóját és metódusát.

Egységbezárás

A fenti megoldás nagyon szép és nagyon jó, de... Ugye sokszor találkozunk ezzel a mondatszerkezettel? Most is ez következik.

A szakácskönyvben mindent publikussá tettünk, azaz mindenki mindenhez hozzáfér. Azonban mi magunk sem szeretnénk, ha minden adatunk a hátunkra lenne írva, hogy az utcán sétálva bárki láthassa életünk minden egyes apró részletét. Szeretnénk, ha lehetnének privát, személyes dolgaink, és olyanok is, amelyeket csak bizonyos személyekkel *(objektumokkal)* osztunk meg.

Ezért lehetőség van arra, hogy a változókat illetve metódusokat ne public láthatósággal adjunk meg, mint ahogy azt a fentiekben tettük. A private kulcsszóval definiált változók és függvények teljes mértékben elzártak lesznek, azaz az adott osztályon kívül senki nem férhet hozzájuk. Az ilyen típusú változók csak az osztályon belül érhetők el, a metódusok pedig csak ott hívhatók.

Mindez arra jó, hogy az osztályok között jól körvonalazott interfészeket definiálhassunk, hogy kizárólag ezeken keresztül férhessenek egymáshoz. Így elérhető, hogy transzparens módon változtathassuk egy osztály belső szerkezetét anélkül, hogy kifelé ebből bármi észrevehető lenne.

Például ha a fenti példában a kalória ábrázolására nem int típusú változót definiálnánk, hanem bevezetnénk egy saját típust, amely a KJ és Kcal értékeket egyaránt tartalmazó rekord lenne, bajban lennénk, hiszen az összes ételt át kellene írni a megfelelő új típusokkal.

Ezzel szemben ha a Kalória attribútum rejtett *(private)*, kívülről nem férhetünk hozzá, nem is látható. Ehelyett hozzuk létre a GetCal és SetCal nevű publikus metódusokat, a következő megfontolásokkal: eddig a KJ értékeket tároltuk el, most vezettük be a Kcal-t. A kettő közötti átjárás úgy történik, hogy a Kcal értéket 4.18-cal kell megszorozni ahhoz, hogy a KJ értéket megkapjuk. Ha az int típusú Kalória változónk rejtett volt, és a Get/Set párral értük el, akkor ezek megvalósítása így nézett ki:

```
public abstract class Étel
{
        private time elkészítésiIdő;
        private int kalória;

        public virtual void Elkészít()
{
/* … */
}
```

```
        public virtual void Tálal()
        {
/* … */
        }

        public virtual int GetCal()
        {
                return this.kalória;
        }

        public virtual void SetCal(int KJvalue)
        {
                this.kalória = KJvalue;
        }
```

A GetCal metódus tehát a Kalória egész értékét adja vissza, a SetCal bemeneti paramétere pedig a beállítandó KJ érték, amelyet egyszerűen hozzárendelünk a Kalóriához.
Ha a struktúrát kibővítjük a Kcal értékek bevezetésével is, kódunk a következőképp fog kinézni:

```
public class CalType {
        public int KJval = 0;
        public int Kcalval = 0;
}

public abstract class Étel
{

        private time elkészítésiIdő;
        private CalType kalória = new CalType();

        public virtual void Elkészít()
{
/* … */
}

        public virtual void Tálal()
        {
        /* … */
        }

        public virtual CalType GetCal()
        {
                return this.kalória;
        }

        public virtual void SetCal(int KJvalue)
        {
                this.kalória.KJval = KJvalue;
        this.kalória.Kcalval =
          (int)(KJvalue / 4.18);
        }
}
```

A SetCal tehát ugyanúgy KJ értéket kap paraméterként, s beállítja mind a KJ, mind a Kcal értékeket. Egyedül annyi a különbség az előző megoldáshoz képest, hogy a GetCal nem int, hanem CalType típusú értéket ad vissza. Ez sajnos azt jelenti, hogy az osztály interfésze megváltozott, kívülről már nem transzparens a belső módosítás.
Ez az a pont, amiért e korai terveket jól és alaposan kell elkészíteni. A fenti kalória egy tipikus olyan példa, amikor a tervezés kezdeti szakaszában nem gondoltuk végig, hogy a változónak milyen típusúnak kell lennie, illetve milyen bővítésekre lehet igény a felhasználó oldaláról.
Másik tipikus, ehhez hasonló példa a pénzértékek tárolása. Első megközelítésben gyakran int vagy long típusú változókat hozunk létre, majd a fejlesztés egy későbbi szakaszában *(rosszabb esetben éles használat közben)* jön rá a felhasználó, hogy jó lenne többféle valutanemet is lekezelni, és egyébként sem egyértelmű, hogy az a 25.000.000 érték forintban vagy svájci frankban értendő. Ekkor létrehozunk egy Money osztályt, ahol a valutanemet és az összeget egyaránt tárolni tudjuk, és máris megváltozott az osztály interfésze…

Alig írtam néhány oldalt, s máris mennyi probléma merült fel. Az esetek többségében a szoftverfejlesztők és –tervezők mind szembesülnek ezekkel a problémákkal: sajnos sokszor saját bőrükön tapasztalják, milyen jó lett volna gondosabban tervezni. A következő hónapokban néhány ilyen buktatót *(és megoldást)* szeretnék bemutatni, amelyeken vagy már túlestem jómagam, vagy láttam a negatív példát, és szeretnék tőle megkímélni mindenkit, ha lehetséges.
Sor kerül tehát a bevezetőben említett valamennyi rövidítés értelmezésére, magyarázatára. Aki pedig nem tudja, mi az a HBR, járjon utána

Molnár Ágnes
aghy@vipmail.hu

Legyen Ön is a nagyok között!

Microsoft CERTIFIED Professional

Kedvezményes hivatalos Microsoft mérnök képzések ősztől!

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.