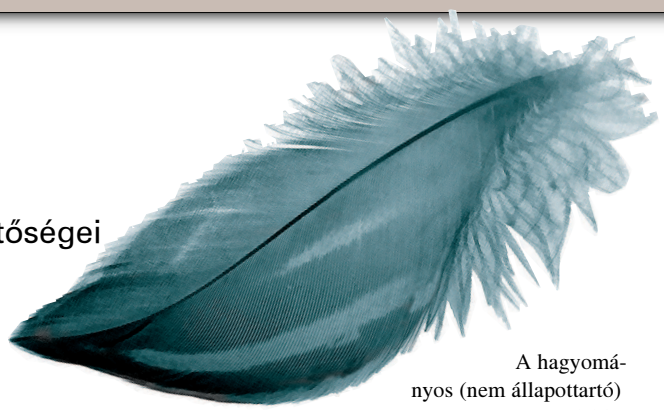
# Könnyű álom (6. rész)

## A 2.4-es rendszermag hálózatvédelmi lehetőségei



A Linux 1994 óta tartalmaz csomagszűrőt. Az elsőt még a méltán híres *Alan Cox* építette be az 1.1-es sorozatba a BSD ipfw-jének alapjain. Ezt fejlesztette tovább *Jos Vos* és még néhányan, így kerül be a 2.0-s rendszermagba. Az emberek legnagyobb része egyszerűen ipfwadm-nek hívja a kezelőprogramja alapján. 1998 közepén a 2.2-höz *Rusty Russell* és *Michael Neuling* teljesen átdolgozták a csomagszűrő alrendszert, és elkészítették az ipchains nevű programot, mellyel a rendszermag beállításait lehet piszkálgatni. A rendszer jó, csak kissé rugalmatlan. Ezt látva Rusty 1999 közepén hozzáfogott a negyedik sorozatú Linux csomagszűrő tervezésének, melyet a 2.3.15-ös fejlesztői rendszermagba építettek be először. Azóta számos kisebb-nagyobb fejlesztésen ment át, és jelenleg a 2.4.4-ac9-ben üzembiztos, jól használható eszközzé vált. Erről szól ez a mese. Leírjuk, hogy milyen lehetőségeket ad a rendszer, hogyan használható. Nem törekszünk azonban a referencia szintű leírásra, mivel több ilyen leírás is létezik (bár ezek a leírások általában nehezen emészthetők).

Mielőtt azonban belekezdenénk a rendszer ismertetésébe, soroljuk fel, hogy kiknek köszönhetjük a rendszer létrejöttét: *Paul „Rusty" Russell*, *Harald Welte*, *James Morris*, *Marc Boucher* (külön öröm, hogy a rendszert folymatosan gyógyítgatják magyar varázslók is: *Kadlecsik József* és *Kis-Szabó András*). Meg kell jegyeznünk még valamit: a netfilter elsődleges célja a csomagszűrés, de nagyon fontos lehetőségei vannak ezen kívül, ami ugyan a biztonság peremterületeire visz minket, de meggyőződésünk szerint segít a biztonságosabb rendszer kialakításában. Mivel a lehetőségek meglehetősen bőségesek, így a sorozat ezen darabja két összefüggő részből fog állni, így teljes értékűnek csak a másikkal együtt tekinthető.

### Néhány alapvető fogalom

A hálózati védelem helyes beállításának alapvető feltétele az, hogy tökéletesen átlássuk a hálózati kapcsolattartás működését, és az adott védelmi rendszer által nyújtott lehetőségeket. A Linux védelmi rendszerének helyes beállításához meg kell ismernünk a kapcsolódó hálózati fogalmakat, és a csomagszűrő rendszer lehetőségeit. Először igyekszünk megvilágítani néhány későbbiekben felmerülő fogalmat. Ezek megértése szükséges ahhoz, hogy valaki helyesen tudjon beállítani egy csomagszűrőt.

### Csomagszűrő

Csomagszűrőnek nevezzük azt a rendszert, amely képes a hálózaton közlekedő csomagokat valamilyen szempontok szerint szétválogatni jó és rossz csomagokra. A szempontokat a rendszer fejlettsége határozza meg. Minden csomagszűrő képes a csomagok feladója és célja alapján válogatni, a jobbak a csomagok több tulajdonságát is megvizsgálják (IP-lehetőségek, TCP-zászlók, ethernet forráscím stb.). A Linux 2.2-es sorozata megbízható, de egy nem túl nagy tudású csomagszűrővel rendelkezik. A régebbi rendszer lehetőségeit fejlesztője szerint sem lehet egyszerűen kiterjeszteni, nem moduláris, bizonyos alapvető szükségletek ellátására nem képes, és még sorolhatnánk. Az új magba fejlesztett rendszer minden területen felülmúlja elődeit. A lehetőségei modularitásának köszönhetően kimeríthetetlenek. Ha valaki hiányol valamilyen lehetőséget, nekiállhat modult fejleszteni rá, de az alapmodulok az egyik legjobban használható csomagszűrővé teszik.

A hagyományos (nem állapottartó) csomagszűrők rossz tulajdonsága, hogy nem képesek a kapcsolatokat egységként kezelni, így olyan csomagokat is átengedhetnek, amelyek szabályosnak látszanak ugyan, de nem tartoznak semmilyen korábban felépített kapcsolathoz. Így hagyományos csomagszűrőn keresztül bizonyos körülmények között fel lehet építeni egy kapcsolatot úgy, hogy a forrás és a célpont nem használ SYN-es csomagot. Ez természetesen csak akkor lehetséges, ha a támadó a teljes IP-vermet lecseréli. Tételezzük fel – csak a vita kedvéért –, hogy támadónk elég elszánt és valóban belepiszkál a védett hálózat egyik gépének IP-vermébe. Ha a hagyományos csomagszűrőnk a külső területről 80-as, azaz a webkapuról érkező csomagot lát, amelyben nincs beállítva a SYN zászló, akkor joggal hiheti, hogy ez már egy élő, bentről kezdeményezett kapcsolathoz tartozó csomag. Mint ilyen, nincs vele semmi gond, így nyugodtan be lehet engedni. Ha a támadó felkészített egy belső rendszert, akkor ezzel a csomaggal akár kapcsolatot is kezdeményezhet, így akkor jár be, amikor csak akar. Mindezt azért teheti, mivel a rendszer minden csomagról különálló egységként dönt. Ha a csomagszűrő meg tudná állapítani, hogy ez a csomag egy még el sem kezdett kapcsolat része, akkor védekezhetne ellene. Ha a csomag kizárólag alkalmazásszűrő tűzfalon (a későbbiekben bővebben lesz szó róla) tudna bejutni a védett hálózatba, akkor a tűzfal azonnal eldobná a csomagot, mivel az nem szabályos.

### Állapottartó csomagszűrő

Az állapottartó csomagszűrő (Stateful Packet Filter – SPF) olyan csomagszűrő rendszer, amely nem elégszik meg azzal, hogy a csomagokat önmagukban minősíti, hanem egy kapcsolattáblában jegyez minden szabályosan felépített kapcsolatot, és így a fenti példában megadott esetben észlelni tudja, hogy a csomagok csak látszólag helyesek, valójában nem tartoznak semmilyen szabályosan felépített kapcsolathoz.

E szűrők már ki tudják szűrni a lopakodó kapupásztázásokat (stealth scan – lásd lent), mivel azok éppen a helyesnek tűnő csomagok elvén próbálnak behatolni a védett hálózatba. Ha a pásztázáshoz különleges csomagokat használnak (Xmas vagy Null), azt egy jobb hagyományos csomagszűrő is ki tudja szűrni. Ha azonban a csomagok magukat felépült kapcsolatnak álcázva igyekeznek bejutni a védett területre, akkor azt az állapottartó csomagszűrő megakadályozza.

Az állapottartó rendszerek apró szépséghibája a hagyományossal szemben az, hogy itt már oda kell figyelni a rendszer esetleges újraindításánál, mivel a már élő kapcsolatok ilyenkor elszállhatnak. Nagy számú kapcsolat esetén előfordulhat, hogy a kapcsolattábla telítődik. Ha a sysctl támogatást befordítjuk a magba, a tábla mérete állítható (/proc/sys/net/ipv4/ip_conntrack_max, vagy szerencsésebb az ip_conntrack modul 'hashsize' értéket módosítani – ip_conntrack_max=hashsize*8), de DoS (vagy DDoS), támadás esetén bármekkora tábla meg fog telni.

A jobb minőségű SPF-ek a csomagok továbbítása közben le tudják cserélni azok azonosítóit (sequence number), így védve a belső hálózat rossz minőségű IP-alrendszerrel rendelkező gépeit. Az alkalmazásszintű tűzfalak ezt szükségképpen tudják, de erről a későbbiekben még szó lesz.

## Kapcsolatok lehetséges számának meghatározása (rate limiting)

Sokan gondolták már, hogy megoldották a DDoS kérdését. Láttam már ilyen magkiegészítést (patch) is. Ez természetesen a támadás jellegéből adódóan nem működhet, hiszen a DDoS esetén már a bejövő csomagok mennyisége is akkora lehet, hogy a hálózat eldugul. Ha DoS-ról van szó, akkor bizonyos esetekben lehetséges a védekezés. Csak a DoS-támadások egy része szűrhető ki, hiszen DoS az is, ha a kiszolgálót össze lehet omlasztani egyetlen jól irányzott kéréssel. Ha egy rendszert a hagyományos módszerrel próbálnak elérhetetlenné tenni, azt meg lehet akadályozni. Hogyan? A kiszolgáló típusától függően jól leírható időminta szerint szolgálja ki ügyfeleit. A webkiszolgálótól egy ügyfél sem kérdez percenként néhány tíznél többször (a töltögető-robotoktól eltekintve, de az már DoS-támadásnak tekinthető). Ha be tudom határolni az ügyfelektől érkező kérések számát valamilyen időszeletben (másodperc, perc, óra, év stb.), akkor a rendszert nem lehet túlterhelni. Nyilván olyan számokat kell választani, hogy a kiszolgáló a legnagyobb terheléssel még működjön, de ne lehessen a terhelést e fölé vinni.

## Kapupásztázás (portscan)

A nyitott kapuk (port) megkeresése valamilyen módszerrel. A hagyományos módszer megpróbált felépíteni egy kapcsolatot a célgép vizsgált kapujával, így érzékelve, hogy az adott kapuban figyel-e valamilyen démon. Ebben az esetben a démonok naplózhatták a kapcsolat felépítésének a tényét, illetve a tűzfalakon az ilyen kapupásztázások nem tudtak keresztüljutni, mert olyan kapcsolatokat akartak felépíteni, amelyek nem voltak engedélyezettek. A későbbiekben egyre inkább a különleges csomagokkal vagy hagyományos csomagokkal, de különleges módon történő kapupásztázás kerül előtérbe. Ezen módszerek alapelve az, hogy egy rendszer nyitott kapuja másként viselkedik egy adott csomagra, mint egy zárt. Például egy SYN-es csomag küldése esetén a nyitott kapu szabályos SYN+ACK csomagot küld vissza, a zárt viszont RST-t, FIN csomag küldésekor a nyitott nem válaszol, a zárt RST-t küld vissza. Ha tehát ezeket a módszereket használjuk, a nyitott kapuk megállapíthatók. Mennyivel előnyösebb ez a támadó számára? Mivel a kapcsolat még nem épült fel, így a rendszer csak külön erre a célra készített eszközzel tudja érzékelni és naplózni, tehát nehezebb észrevenni. Előnye továbbá, hogy mivel nem feltétlenül kell SYN-es csomagot küldeni a kapupásztázáshoz, így bizonyos tiszta csomagszűrőkön át tud hatolni. Mivel azonban az új mag csomagszűrőjét felszerelék kapcsolatkövető (connection tracking) rendszerrel, így helyes használatával az ilyen pásztázások kivédhetők (Fjodornak, az nmap írójának őszinte sajnálatára).

## MAC cím

Az ethernethálózatok belső címzési rendszere a MAC címen alapul. Elvileg minden ethernetprotokollt ismerő eszköznek egyedi címe van. Kivéve az ötödrangú tajvani holmikat, ahol a gyártó költségcsökkentés miatt azonos címeket használt… Ha azt akarjuk, hogy az IP-cím átvételével még ne lehessen átvenni valakinek (vagy adott gépnek) a jogosultságait (hiszen a tiszta csomagszűrők szótárából hiányzik a felhasználó kifejezés), beállíthatjuk, hogy kizárólag egy adott MAC-cím tehessen valamit. Ez nem túlzottan erős feltétel, mivel szinte minden operációs rendszer lehetővé teszi a MAC cím programból történő állítását, de azért ez is valami. Az, hogy ezt a hálózati kártyák támogatják, a Decnet protokollcsaládnak köszönhető. Köszönjük szépen, Digital!

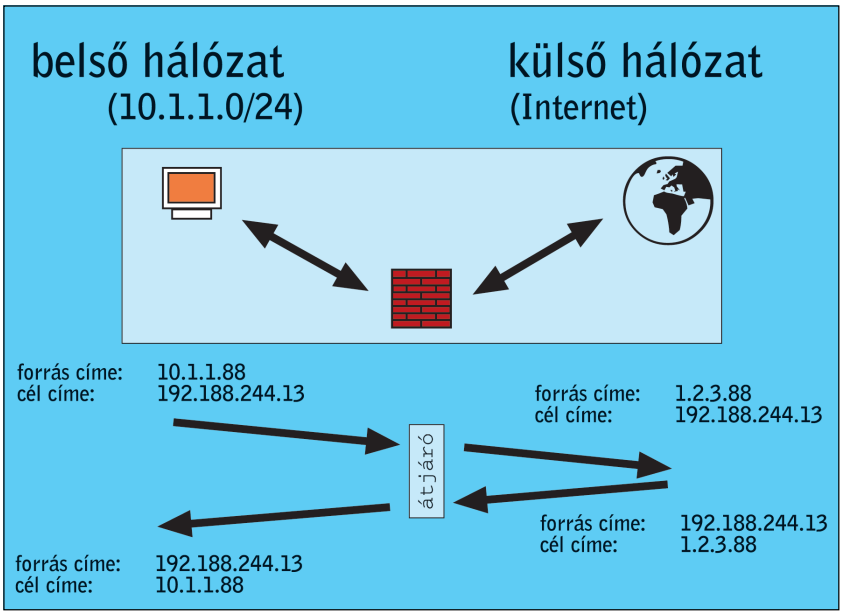
## Hálózati címváltás

A hálózati címváltás (Network Address Translation – NAT) azt jelenti, hogy az útválasztó két hálózat között úgy viszi át a csomagokat, hogy közben azok forrás-, vagy célcímét módosítja. A NAT folyamata:

- a csomag IP-fejlécében a cél-, illetve forráscímek szükség esetén átíródnak,
- a csomag UDP – vagy TCP – fejlécében a kapucímek szükség esetén átíródnak (PAT, lásd később),
- ha a kapuk megváltoztak, a TCP-fejléc ellenőrzőösszeg mezője újraszámítódik,
- az IP-csomag ellenőrzőösszeg mezője újraszámítódik,
- a rendszer további jó utat kíván a csomagnak.

Ha a csomag valamelyik jellemzőjét megváltoztatjuk, hogyan talál vissza az eredeti feladóhoz, vagy hogyan talál oda a címzettjéhez? Úgy, hogy a címváltó a két hálózat között helyezkedik el, és az egyik irányba menő csomagok módosításait visszafordítja. Ha például a belső hálózatunk címe 10.1.1.0/24, és ezt fordítjuk Interneten is használható 1.2.3.0/24 (Ez csak példa!) címtartománnyá, akkor a folyamat a következőképpen zajlik (*1. ábra*):

- a kimenő csomag megérkezik az útválasztóhoz, az észleli, hogy a csomag 10.1.1.88-as forráscímmel igyekszik kijutni a hálózatból,




*1. ábra* Hálózati címváltás (NAT)

- megállítja és kicseréli a forráscímét 1.2.3.88-ra,
- továbbküldi a külső hálózati csatolón keresztül,
- a válaszcsomag az 1.2.3.88-as célcímre fog érkezni a külső láb felől, ezt az útválasztó észleli,
- a csomagot megállítja, a célcímet lecseréli, és továbbítja a belső láb felé és így tovább…

A NAT szükségességének oka lehet például, hogy az egyik hálózat felől a másikat egyetlen címűnek szeretnénk láttatni (álcázás, átirányítás, terhelés-elosztás), vagy a hálózatok valós címtartományát kell eltolni (például mindkét hálózatnak 10.1.1.0/24 címe van). Elmondható, hogy a NAT leggyakrabban IP-címeket [1.] alakít külső hálóza-

ton is használhatóvá, e címeket ugyanis az útválasztók nem továbbíthatják az Internetre. Így ilyen magánhálózatok kívülről való elérését a legegyszerűbb valamilyen címfordító eszköz segítségével megoldani. Lehetséges alkalmazásszintű átjáróval is (később lesz róla szó), de az szintén NAT feladatot lát el. A továbbiakban a következő szakszavakat fogjuk használni: a belső (local) cím cserélődik ki egy külső (global) címre és viszont. Lehet sok–sok és sok–egy típusú NAT. Alapvetően kétféle NAT létezik: statikus és a dinamikus. Ezeket a kifejezéseket használják a kapcsolódó témájú RFC-k [2.][3.], de mivel a fejlesztőknek ez valamiért nem tetszett, a netfilter leírásai más felosztást használnak. Először megmutatjuk, hogy mit jelent a statikus és dinamikus fogalom ebben az esetben, a későbbiekben ezt megfeleltetjük a netfilter fogalmainak.

### Statikus NAT

Ahogy a neve is mutatja, merev egy az egyes címmegfeleltetést jelent a külső és a belső címtartományok között. A fordítandó címtartománynak legalább annyi címmel kell rendelkeznie, amennyi az aktív belső címek száma.

### Dinamikus NAT

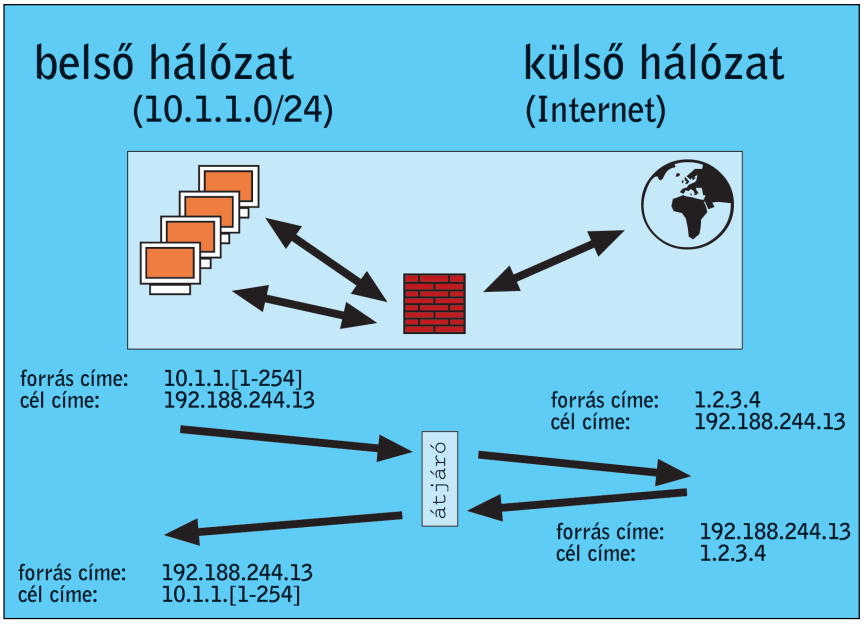
A belső címek csoportját fordítja a külső címek csoportjára. Itt csak az a kitétel, hogy legalább annyi cím legyen a fordítandó címtartományban, ahány az egyszerre működő belső címek száma.

### Kapuváltás

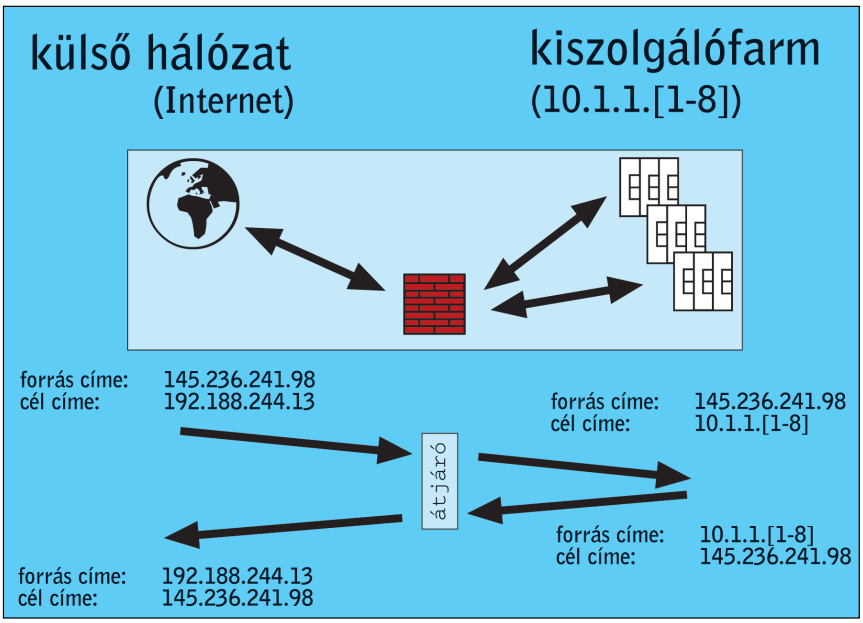
A kapuváltás (Port Address Translation – PAT) azt jelenti, hogy a PAT-ot végző eszköz a TCP-, vagy UDP-csomagok forrás-, illetve célkapuit módosítja.

### DNAT IP-tömörítéssel

A PAT-ra például akkor lehet szükség, ha nem áll rendelkezésre annyi külső cím, amennyi az egyszerre működő belső címek száma. Itt a PAT-ot végző eszköz általában a forrás címét és kapuját módosítja, így lehetővé válik több belső címről érkező kapcsolat akár egyetlen külső címen láttatása *(2. ábra)*. Ilyen eset a 2.2-es rendszermag jól ismert álcázás (masquerade) lehetősége.




*2. ábra* Álcázás (DNAT IP-tömörítéssel)




*3. ábra* Terheléselosztás (DNAT, PAT)

### DNAT külső csatoló többszörözéssel (Interface Redundancy)

Ha a rendszernek több külső csatolója van, akkor lehetséges, hogy a belső címeket a külső címek olyan csoportjaira képezzük le, amelyek lehetővé teszik, hogy kihasználjuk a több külső csatoló adta nagyobb sávszélességet. Ilyen eset lehet, ha több internetes kapcsolatunk van, és szükség esetén növelni akarjuk a sávszélességet.

### Kapcsolat eltérítése (redirection) SNAT + PAT

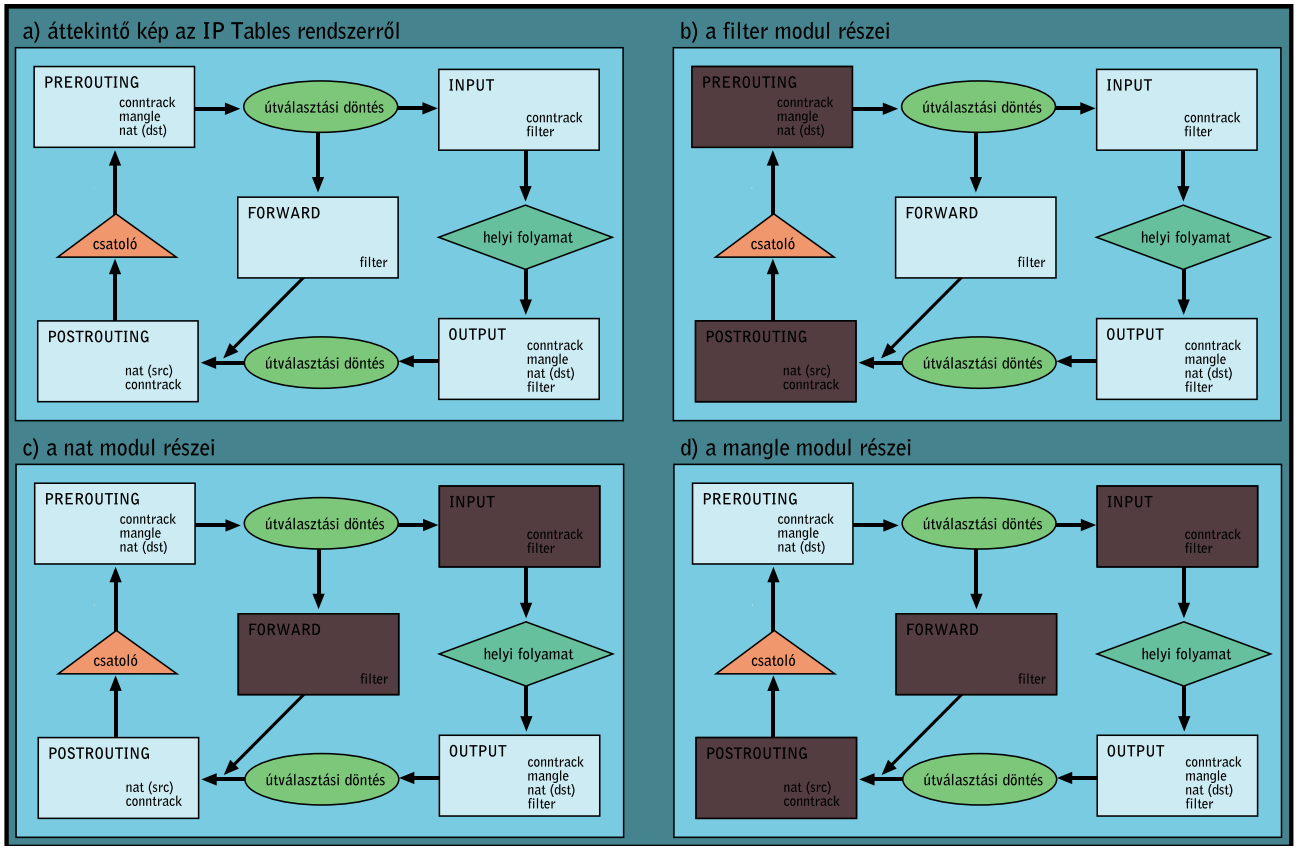
Előfordulhat, hogy a kapcsolatokat az eredeti céltól eltérő irányba kell továbbítani. Ilyen eset lehet, ha a rendszer egyetlen valós, külső hálózati címén szeretnénk több szolgáltatást is láttatni, melyek valójában több gépen futnak.

### Terheléselosztás

A terheléselosztás (Load balance) különleges kapcsolateltérítés, ahol a cél több gép. Így egy erősen terhelt szolgáltatást egy kiszolgálócsoport visz oly módon, hogy a külvilág számára azok mindegyike egyetlen címen látszik *(3. ábra)*.

## Átlátszó proxy

Az átlátszó proxy (Transparent proxy) korszerű hálózati határvédelmi eszközök által támogatott módszer. Azt jelenti, hogy a belső hálózat gépei számára nem látható, köztük és az igénybe vett kiszolgáló között bármilyen kapcsolatellenőrző eszköz lehet. Használata esetén nincs szükség arra, hogy az ügyfélprogram támogassa a tűzfalon keresztül történő kapcsolattartást. A cikksorozat negyedik részében erről már bővebben írtunk.




*4. ábra* Az IP Tables rendszer és annak részei

## A netfilter működésének alapelvei

A netfilter működési elve a Unix filozófiáját követi. Modulokból áll, amelyek célja a hálózaton közlekedő csomagok módosítása. Ez azt jelenti, hogy módosíthatók a csomagok jellemzői, szükség esetén a rendszer bele is szólhat a kapcsolatba. Mivel a netfilter nem más, mint általános keretrendszer csomagok módosítására, így elvileg tetszőleges hálózati protokollal tud dolgozni. Jelenleg még csak az IPV4 protokollt támogató modulok vannak készen és bizonyos részek az IPV6 kezelésére, de elvileg lehetőség van akár IPX vagy NetBEUI szűrők elkészítésére is. Más kérdés, hogy ezek feltehetőleg soha nem készülnek el. Az IP-kezelő alrendszer az IP Tables és a hozzá tartozó alkalmazásszintű eszköz neve iptables.

Az IP Tables szűrőkből (chain) áll, ezek a szűrők szabályokat tartalmaznak, melyek egy feltételből és egy műveletből állnak. A feltétel lehet összetett, és a csomag valamely jellemzőjét vagy kapcsolatának állapotát vizsgálja. A hozzá rendelt műveletben pedig meghatározzuk, hogy a feltétel teljesülése esetén mit csináljon a rendszer.

Az IP Tables felépítése a *4. ábrán* látható. A rendszer teljesen moduláris, így annak egyes részei csak akkor élnek, ha arra szükség van. Az ábrán a rendszer egyes részein feltüntettük, hogy melyik modulnak részei. A teljes rendszerben a csomag életútja a következő módon néz ki:

A hálózati csatolókról bejövő csomagok először a PREROUTING szűrőbe esnek be, ezután a rendszermag hálózati útválasztója eldönti, hogy a csomag helyi vagy továbbítandó (esetleg nem továbbítható – ebben az esetben azonban „network unreachable icmp” csomagot küld vissza). Ha a csomag *helyi*, akkor az INPUT szűrőn keresztül helyi folyamathoz kerül a benne lévő adat. Ha *továbbítandó* és a csomagtovábbítás engedélyezett (ip_forward), akkor a FORWARD szűrőn keresztül halad tovább és a POSTROUTING-on át jut a hálózati csatolóra. A helyi folyamatok által feladott csomagok az OUTPUT után szintén a POSTROUTING szűrőn át kerülnek a csatolókra. A pontosság kedvéért el kell mondanunk, hogy ezek a szűrők nem mindig állnak összefüggésben egymással. A rendszer bizonyos szolgáltatásai használják, mások pedig nem. Az ábra arra fekteti a hangsúlyt, hogy áttekinttő képet adjon, az egyes tábláknak csak bizonyos szűrők részei, de ha látjuk, miként mozog a csomag a rendszerben, akkor érteni fogjuk a szűrők egymásra gyakorolt hatását is.

A rendszer jelenleg három alapvető részt tartalmaz: a filter, a NAT és a mangle modulokat. Minden modul tartalmaz bizonyos szűrőket (előfordul átfedés is), amelyekbe a szabályokat a felhasználó beilleszthet. A szabályok feltételei szinte korlátozás nélkül használhatók (ilyen szabály lehet például az, hogy mi egy csomag forráscíme) minden szűrőben. Azt, hogy egy-egy szűrő melyik alrendszerhez tartozik, az határozza meg, hogy milyen műveletet rendelhetünk az adott szabályhoz. Míg csomagot eldobni (DROP) minden szűrőben lehet, addig a csomag célját megváltoztatni (REDIRECT) kizárólag a NAT-modul által használt PREROUTING és OUTPUT szűrőkben lehetséges.

A szűrő alrendszer tiszta csomagszűrő rendszer kifinomult szűrési lehetőségekkel. A 2.2-es sorozatú rendszermagokban lévő csomagszűrő (ipchains) hasonló tulajdonságokkal bírt, bár a források teljes újraírása mellett sok tulajdonság megerősödött. Az INPUT, a FOWARD és az OUTPUT beépített szűrőket tartalmazza. Célja a bejövő, áthaladó és kimenő csomagok szűrése.

A NAT-alrendszer célja a hálózati címfordítás megvalósítása. A szűrői olyan műveleteket támogatnak, amelyek segítségével a csomagok forrás-, és célcímei módosíthatók.

A mangle-alrendszer általános támogatást nyújt az útválasztás előtti csomagmódosításhoz. A csomagokat megjelölhetjük, ezt az útválasztók a későbbiekben figyelembe veszik.

## Az IP Tables szűrőinek (chain) működése

A szűrőkben szabályok érvényesülnek. A szabályok felülről lefelé kiértékelődnek, és egy csomag vizsgálata addig tart, amíg valamelyik szabályban megadott feltétel nem bizonyul igaznak a csomagra nézve, és a szabályban meghatározott művelet (TARGET) nem utasítja megállásra a szűrőt. A kijelentésből látszik, hogy kétféle művelet lehetséges: egy amelyik lezárja a vizsgálatot, és egy amelyik nem. A működése példán keresztül érthetőbbé válik. Adott egy csomag, a 10.1.1.1-es (a továbbiakban egyes) címről tart a 10.1.2.2-es (a továbbiakban kettes) címre. Ha a rendszerünk a két gép között tűzfalszolgáltatást lát el, akkor a korábbiakban vázolt minta szerint a csomag áthalad a FORWARD szűrőn. A szűrőben az alábbi szabályok érvényesek:

1. szabály >> feltétel: a csomag forrása a 10.1.1.5-ös gép?
   művelet: továbbítás elutasítva
2. szabály >> feltétel: a csomag forrása a 10.1.1.0-s hálózat?
   művelet: naplózni
3. szabály >> feltétel: a csomag forrása a 10.1.1.0-s hálózat?
   művelet: továbbítás engedélyezve
4. szabály >> feltétel: nincs feltétel
   művelet: naplózni
5. szabály >> feltétel: nincs feltétel
   művelet: továbbítás elutasítva

A fenti szabályrendszer esetén az adott csomaggal a következő történik: az első szabály által megadott feltételnek nem felel meg, így a szabály művelete sem hajtódik végre. A második szabály illik rá. A szabályhoz tartozó utasítás megkéri a rendszert, hogy a csomag jellemzőit a rendszer naplóállományában rögzítse (LOG művelet). Ez az művelet nem állítja meg a csomag további vizsgálatát, mivel arról nem hozott egyértelmű döntést, hogy az továbbítható-e vagy sem. A harmadik szabály ugyanazzal a feltétellel már világos döntést hoz a csomag sorsáról: továbbítható (ACCEPT művelet). Amennyiben a csomag jellemzői nem felelnek meg egyik korábbi feltételnek sem, akkor az utolsó szabály célszerűen az, hogy minden más csomagot naplózás után elutasít (DROP vagy REJECT művelet). Ennek jelentőségéről bővebben írtunk a cikksorozat negyedik részében. Mivel a szűrőnek minden esetben valamilyen világos művelettel kell végződnie, így minden beépített szűrő rendelkezik egy végső művelettel, amely meghatározza, hogy mi történjen az addig le nem kezelt csomagokkal. Ezt hívják a szűrő alapértelmezett szabályának (default policy).

Felmerülhet a kérdés, miért fogalmaztunk úgy: minden beépített szűrő. Talán van valami egyéb is? Nos igen. Az IP Tables örökölte az ipchains egyik nagyon hasznos tulajdonságát: a felhasználó is meghatározhat szűrőket, és szükség esetén a csomagot átadhatja ennek a szűrőnek vizsgálatra. A felhasználói szűrőknek nem lehet alapértelmezett szabálya, ha a csomagra egyik szabály feltételei sem illenek, akkor a csomag ellenőrzése a felhasználói szűrőt meghívó utáni szabállyal folytatódik. Ha egy ilyen szűrőből vissza szeretnénk térni a meghívó szűrőbe, akkor a RETURN célt kell használnunk. Lehetséges újabb saját szűrők meghívása akár egymásba ágyazva is, de ha ciklus alakul ki (tehát a csomag visszaér a kiindulópontra), akkor a rendszer eldobja a csomagot.

A szűrőket a következőképpen célszerű használni: eldöntjük, hogy milyen feltételekkel szabad a szűrőnek átengednie a csomagot. A feltételeket a később ismertetett formában átírjuk IP Tables által is értelmezhető feltételekké. Amennyiben valamennyi feltételünk teljesül, a csomagot elfogadjuk, vagy visszatérünk a hívó szűrőbe. Ha a csomagra nem teljesül egyik feltételünk sem, akkor naplózzuk és elutasítjuk. A szűrők utolsó szabálya tehát általában feltétel nélküli tiltás. Amennyiben nem így teszünk, akkor az esetleges feledékenység (elfelejtünk tiltani valamilyen forgalmat) szivárgást idézhet elő.

A beépített szűrők alapértelmezett szabálya csak ACCEPT vagy DROP lehet. Ha egy beépített szűrőben RETURN műveletet hajtunk végre, az alapértelmezett szabály lép életbe.

## Az IP Tables által értelmezett feltételek

A rendszer lehetővé teszi nagyon finoman hangolható szabályok leírását. Az alábbiakban röviden bemutatjuk, milyen feltételek alapján vizsgálhatunk meg egy csomagot. Ahol megadható a feltételben felkiáltójel, ott a feltétel tagadását jelenti.

Tehát a -s 10.1.1.1 azt jelenti, hogy a csomag forrása a 10.1.1.1 című gép, míg a -s ! 10.1.1.1 azt, hogy a forrás nem a 10.1.1.1. A feltételek együtt is használhatók, ekkor a feltétel az egyes alfeltételek logikai *és* kapcsolata. Így ha adott a következő feltétel: -s 10.1.1.1 -p TCP, az azt jelenti, hogy a csomag TCP protokollt tartalmaz és a forráscíme 10.1.1.1. A rendszer kétféle feltételt különböztet meg. A belső feltételek modul betöltése nélkül használhatók, a kiterjesztett feltételeket csak a megfelelő modul betöltése után használhatjuk.

### Belső feltételek

-p | --protocol [!] protokoll

Megadhatjuk vele, hogy a csomag IP-kerete milyen protokollazonosítót tartalmazhat (erről a sorozat harmadik részében olvashattak).

-s | --source [!] cím[/maszk]

-d | --destination [!] cím[/maszk]

Segítségével a csomag forrás- és célcímét határozhatjuk meg.

-i | --in-interface [!] [csatolónév]

-o | --out-interface [!] [csatolónév]

Megvizsgálhatjuk, hogy a csomag mely csatolón jött be, és melyiken fog távozni. Amennyiben a csatoló nevének végén '+' van, akkor minden olyan csatolóra illeszkedni fog a feltétel, amelynek így kezdődik a neve.

[!] -f | --fragment

Igaz, ha a csomag töredezett és nem az első töredék. Ha a "!" is meg van adva, akkor a töredezett csomagnak csak az első darabjára és a nem töredezett csomagokra ad vissza igazat. Kapcsolatkövetés (connection tracking) esetén – amit a nat használata automatikusan bekapcsol –nincs értelme, mert a rendszer ilyen esetekben töredezettségmentesíti a csomagokat.

### Kiterjesztett feltételek

Ezeket a feltételeket általában modulok tartalmazzák, melyeket használatuk előtt be kell tölteni. Ez kétféleképpen lehetséges: közvetlenül a "--match" kapcsoló után megadott modulnévvel, vagy közvetett úton, a protokoll meghatározásával (--protocol kapcsoló), mivel ilyenkor az adott protokollra használható feltételek használhatók. Az alábbi felsorolásban először a tartalmazó modul nevét adjuk meg, utána a használható feltételeket.

**tcp modul:** TCP protokoll esetén alkalmazható

--source-port [!] [port[:port]]

--destination-port [!] [port[:port]]

A forrás és a cél TCP-kapuját lehet behatárolni vele. Amennyiben két kapu is meg van adva kettősponttal elválasztva, akkor a két szám között lévő (beleértve a két szélsőt is) kapukra illik majd a feltétel. Használható még a _--sport_ és a _--dport_ rövidítés is.

--tcp-flags [!] mask comp

Igaz, ha a csomag TCP lehetőségei a meghatározott mintának megfelelnek. Mindkét érték a következőkből áll:
SYN ACK FIN RST URG PSH ALL NONE. Az első érték megadja, a megvizsgálandó zászókat, a második pedig meghatározza, hogy a megadottak közül melyiknek kell beállítva lenni.

[!] --syn

Csak azokra a TCP-csomagokra igaz, melyekben a SYN bit be van állítva, de a FIN és az ACK nincs. Ezek azok a csomagok, melyek a TCP-kapcsolat felépítését kezdeményezik. Ha ezeket a csomagokat egy bizonyos hálózatból kitiltjuk, akkor oda hagyományos módon nem lehet kapcsolatot felépíteni. Más kérdés, hogy készíthető olyan trükkös eszköz, amely így is megoldja. Használata egyenértékű az előzőleg említett lehetőség következő használatával:

```
--tcp-flags SYN,RST,ACK SYN
```

--tcp-option [!] number
Akkor igaz, ha az adott TCP-lehetőség (option) be van állítva.

**udp modul:** UDP protokoll esetén
--source-port [!] [port[:port]]
--destination-port [!] [port[:port]]
Az UDP-csomag forrás-, és célkapuját határozhatjuk meg vele. Ha két kaput adunk meg, akkor ugyanúgy viselkedik, mint a TCP hasonló lehetősége.

**icmp modul:** icmp protokollt vivő csomagokra
--icmp-type [!] typename
Az icmp csomag típusát határozhatjuk meg vele. A típus megadható számmal, vagy a típusnévvel, amit az `iptables -p icmp -h` parancs listáz ki.

**mac modul:** a csomagok ethernetcímének meghatározására
--mac-source [!] address
Igaz, ha a csomag ethernetforrása az adott cím. A cím megadásának formátuma a következő: XX:XX:XX:XX:XX:XX, ahol minden X egy hexadecimális számjegy. Csak a PREROUTING, FORWARD vagy az INPUT szűrőkben van értelme.

**limit modul:** egy szabály illeszkedésének lehetséges száma szabályozható a segítségével. Mivel működése viszonylag összetett, és a rendszerleírása [5.] részletesen ismerteti, így annak részletes leírásától itt eltekintünk.
--limit rate
Adott időtartományban meghatározza a lehetséges illeszkedések számát. A tartomány lehet „second”, „minute”, „hour”, „day”.
--limit-burst number

**multiport modul**: A feltételben több kapu megadására nyújt lehetőséget. (Kevesebb, mint 16.)
--source-port [port[,port]]
--destination-port [port[,port]]
A forrás- vagy célkapuk meghatározására használható. Több nem összefüggő kapu is megadható.
--port [port[,port]]
Akkor illeszkedik, ha a forrás- és célkapu egyenlő valamint a felsoroltak között van.

**mark modul:** a korábban jelölt csomagok felismerésére.
--mark value[/mask]
Azokra a csomagokra illeszkedik, melyek a megadott módon vannak jelölve. Ha a mask meg van adva, akkor a jelölés logikai *és* művelet útján kerül csak összehasonlításra.

**owner modul:** a helyben keletkezett csomagok létrehozóját lehet ellenőrizni vele. Csak az OUTPUT szűrőben használható.
--uid-owner userid
--gid-owner groupid
Illeszkedik, ha a csomagot a megadott felhasználói vagy csoportazonosítóval rendelkező folyamat hozta létre.
--pid-owner processid
Illeszkedik, ha a kibocsátó folyamat azonosítója a megadott.
--sid-owner sessionid
Illeszkedik, ha a csomagot létrehozó folyamat a megadott munkamenet-azonosítóval rendelkezik.

**state modul:** a modul a kapcsolatkövetéssel (connection tracking) együtt használva a csomag állapotát határozza meg annak kapcsolatának ismeretében.
--state state[,state]
Illeszkedik, ha a csomag állapota a megadott listában szerepel. Érvényes állapotok: INVALID – nincs ismert kapcsolat, ESTABLISHED – ismert kapcsolat része, NEW – új kapcsolatot kezdeményez, RELATED – valamilyen már felépült kapcsolathoz tartozik (például FTP) vagy egy kapcsolathoz tartozó ICMP üzenet.

**unclean modul:** nincsenek választható lehetőségei. A hibás csomagokra illeszkedik.

**tos modul:** az IP-fejléc Type of Service mezőjének ellenőrzésére.
--tos tos
A szolgáltatás típusának meghatározására. A típus megadható számmal, vagy névvel (az `iptables -m tos -h` parancs kiírja).

**ttl modul:** az IP-fejléc ttl mezőjének ellenőrzésére.
--ttl ttl
Illeszkedik a megadott ttl értékre.

**tcpmss modul:** a TCP SYN-es csomagok MSS mezőjének vizsgálatát teszi lehetővé, ez határozza meg a kapcsolat során használható lehető legnagyobb csomagméretet.

A cikk folytatása a Linuxvilág következő számában lesz olvasható.

## Irodalomjegyzék


[1.] (RFC 1918) Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear: Address Allocation for Private Internets (1996. február)
[2.] (RFC 2663) P. Srisuresh, M. Holdrege: IP Network Address Translator (NAT) Terminology and Considerations (1999. augusztus)
[3.] (RFC 2993) T. Hain: Architectural Implications of NAT (2000. november)
[4.] (RFC 3022) P. Srisuresh, K. Egevang: Traditional IP Network Address Translator (Traditional NAT) (2001. január)
[5.] Paul 'Rusty' Russel: IPTABLES-HOWTO ➲ http://netfilter.samba.org/



*Mátó Péter* (atya@andrews.hu), informatikus mérnök és tanár. Biztonsági rendszerek ellenőrzésével és telepítésével, valamint oktatással foglalkozik. 1995-ben találkozott először linuxos rendszerrel. Ha teheti, kirándul vagy olvas.



*Borbély Zoltán* (bozo@andrews.hu), okleveles mérnök-informatikus. Főként Linuxon futó számítógépes biztonsági rendszerek tervezésével és fejlesztésével foglalkozik. A 1.0.9-es rendszermag ideje óta linuxozik. Szabadidejét barátaival tölti.


*A főszerkesztő ezúton kér elnézést a Tisztelt Olvasótól és a Szerzőktől, ha úgy érzik, hogy a szerzők „technicus terminusainak” magyarítása csorbította a szövegek érthetőségét.*