A Netfilter megszelídítése

Meg kellene erősíteni várkastélyunk árkait? Nézzük meg, hogyan használhatjuk a Netfiltert és az IP Chainst otthoni rendszerünk biztonságának növelésére.

Gratulálok azoknak, akik a mélyvízbe fejest ugorván a 2.2.x (netán 2.0.x) rendszermagról a 2.4.x-re váltottak. Ha mi, akárcsak sokan mások, valamilyen tűzfalat futtatunk az ***IP Chains*** vagy az ***ipfwadm*** segítségével, parancsfájljaink vélhetően egyelőre megfelelőek. Ám előbb vagy utóbb valószínűleg tovább szeretnénk lépni.

A 2.4.x változatszámú rendszermagokba ***Rusty Russell***, a Linux csomagszűrő guruja és programozócsapata beépítette a Netfiltert, ami az IP Chains vagy ipfwadm kiváltására készült. Szerencsére a Netfilter továbbra is lehetővé teszi, hogy IP Chainst vagy `ipfwadm`-ot használjunk, amíg meg nem szokjuk az ***IP Tables*** használatát. Mindezt egy modulba épített rétegen keresztül valósítja meg. A Netfilter azonban olyan sok érdekes újítást tartalmaz, hogy valószínűleg mi is minél előbb a szabályok átalakításán leszünk. Egy dologra viszont oda kell figyelnünk: amennyiben az `ipchains` vagy `ipfwadm` modulokat betöltjük, ezt nem tehetjük meg az `ip_tables`-szel (és vice versa). Szóval mindent vagy semmit. Mindenesetre ennek a cikknek az elolvasása után a változtatások elvégzése magától értetődő lesz.

Azok, akik csak most ismerkednek a csomagszűréssel, nyugodtan hagyják ki az IP Chains-fordításokat, és csak az IP Tables-példákkal ismerkedjenek. Bár nem fogjuk megadni az összes IP Chains parancs és lehetőség IP Tables-fordítását, de írásunk talán jó ötleteket adhat, ha a csomagszűrő tűzfal összeállításához IP Chains parancsokat szeretnénk IP Tables parancsokká alakítani.

Az egyik ok, amiért érdemes Netfilterre váltani, az, hogy (az IP Chainsszel vagy az ipfwadm-mal ellentétben) állapotfüggő (stateful). Hogy ez mit jelent? Azt, hogy képes nyomon követni a kapcsolatokat és engedélyezni a kimenő kérelmekre érkező bejövő válaszokat anélkül, hogy eközben réseket ütne a tűzfalon. A kapcsolat követése itt mindig csak egy bizonyos, ideiglenes lyukat készít a válasz számára, amit kizárólag az adott kiszolgáló használhat. Nemsokára megismerhetjük a működését. A dolog hátulütője, hogy a kapcsolatkövetés ideje alatt a Netfilter egy kicsit több memóriát fogyaszt, mivel a kapcsolatokat a memóriában követi nyomon. Így elképzelhető, hogy 4 MB memóriájú 386-16 gépünk (a szűrési követelmények függvényében) már nem lesz elegendő a feladathoz.

Háttér

A jelenlegi Netfilter-megvalósítás két részre bomlik: a Netfilterként ismert rendszermagrészre, illetve a felhasználói eszköztárra, amely kapcsolatot tart a Netfilterrel, és elkészíti a szabálygyűjteményeket, valamint az IP Tables-adatokat. A csomagszűrő tűzfalhoz mindkettőre szükségünk lesz.

Először összpontosítsunk a rendszermagrészre. A Netfilter támogatja az IPv4 és IPv6 protokollokat, ugyanakkor másféléket nem képes szűrni, ezért tűzfalunk nem futtathat IPX, AppleTalk vagy más efféle protokollt, amit esetleg IP Tables-szabályok kijátszására fel lehetne használni. Továbbá nem szabad engedélyeznünk a gyorskapcsolás (Fast-switching) lehetőséget sem. Ezt a kapcsolót a rendszermag ***Beállítás*** menüjében hálózati lehetőség almenüjének vége felé találjuk. A kód a gyorskapcsolást az IP-verem alacsony szintjén engedélyezi. A Netfilter-kód ennél sokkal magasabb szinten helyezkedik el, így a gyorskapcsolás tulajdonképpen egyszerűen megkerüli őt.

A rendszermag beállítása

Ahhoz, hogy elkezdhessünk dolgozni a Netfilterrel, először is a rendszermagot Netfilter-támogatással le kell fordítanunk. A legtöbb terjesztés alapértelmezés szerint támogatja, de előbb tegyünk egy rövid próbát. Ha az `ip_tables` modult be tudjuk tölteni, ezzel a fejezettel nem kell foglalkoznunk. Rendszergazdaként futtassuk a következő parancsot:

```
modprobe ip_tables
```

Ezután gépeljük be:

```
lsmod | grep ip_tables
```

Ha az `ip_tables` feltűnik, rendben vagyunk. Ha nem, akkor sem kell megijednünk, a rendszermag újrafordítása rendkívül egyszerű. Írásunk ugyan nem fogja ismertetni a teljes újrafordítási folyamatot, de számos forrást találhatunk, ami segíthet bennünket ebben a lépésen. Ha kiderül, hogy rendszermagunkat újra kell fordítanunk, a következő oldalon lévő széljegyzet nyújthat segítséget abban, hogy mit is illesszünk be a rendszermagba.

Netfilter-modulok

Ha az összes modult lefordítottuk és feltelepítettük, valamennyi önműködően be fog töltődni, ha beviszünk valamilyen szabályt, kivéve az `ip_tables`, `ip_nat_ftp` és `ip_conntrack_ftp` csomagokat. Ezeket vagy kézzel vagy IP Tables indító parancsfájlunk részeként tölthetjük be.

A Netfilter teljes fordítása és telepítése rengeteg modult készít, amiből egy átlagos tűzfal azonban csak keveset használ. Azok a modulok, amelyek nem töltődnek be, természetesen memóriát sem fogyasztanak, így nem kell aggódnunk miattuk.

Az IP Tables beszerzése és telepítése

Elképzelhető, hogy terjesztésünk már eleve tartalmazza az IP Tablest, sőt, amennyiben a rendszermag Netfilter-támogatással bír, ez majdhogynem biztosra vehető. Amennyiben a legfrissebb változatot szeretnénk, a Netfilter honlapjáról letölthetjük (➲ http://netfilter.filewatcher.org). Töltsük le és telepítsük az ***INSTALL*** fájl utasításainak megfelelően. A következőkben feltételezzük, hogy a magforrások az ***/usr/src/linux*** könyvtárban találhatók. Ha mégsem, a következő utasításokat a könyvtárnak megfelelően módosítsuk. Amennyiben esetleg a

```
make pending-patches KERNEL_DIR=/usr/src/linux
```

vagy a

```
make patch-o-matic KERNEL_DIR=/usr/src/linux
```

parancsokat le kell futtatnunk, akkor folytatás előtt újra kell fordítanunk a rendszermagot. Egyébként a fenti két parancsot figyelmen kívül is hagyhatjuk. A `patch-o-matic` többnyire

különleges igényű felhasználóknak készült, és nemigen tartozik az átlagos felhasználók érdeklődési körébe.

Futtatás után

```
make KERNEL_DIR=/usr/src/linux
```

majd indítsuk a következő parancsot:

```
make install KERNEL_DIR=/usr/src/linux
```

Most már készen állunk az IP Tables használatára.

Az IP Tables parancssor

Az IP Tables parancssor hat részre bontható. Az első rész az `iptables` parancs, amit a későbbiekben részletezünk; a második a táblameghatározás, a harmadik pedig a láncnév. A negyedik rész a szabálymegadás, amely az IP- vagy ICMP-fejlécek között kereső parancs része, de néhány esetben szabályszám is lehet. Az ötödik rész a cél, végül a hatodik a célérték. Az általános parancssor tehát a következőképpen néz ki:

```
iptables [-t table] -ACDI CHAIN rule-spec
➥ -j TARGET [target option]
```

A fenti sor ugyan nem minden varázslatra igaz, viszont elég általános. A `-L` kapcsolót is igen hasznosnak találhatjuk majd (később még szó lesz róla a cikkben néhány más parancssor-változatról).

Táblák és láncok

A Netfilter három számunkra érdekes táblát tartalmaz. Ezek: a `filter` (szűrő), a `nat` (hálózaticím-átalakítás) és a `mangle` (ferdítő) táblák. Cikkünkben valahányszor csak valamilyen IP Tables-varázslatot mutatunk be, mindig megadjuk a hozzá tartozó táblát is. Egyébként – amennyiben nem adunk meg táblát – a `filter` tábla az alapértelmezett. Éppen ezért, ha nem adunk meg táblát és a szabály hibát jelez, adjunk meg táblameghatározást és próbáljuk meg még egyszer.

Minden táblához egy bizonyos lánc tartozik. A felhasználó által készített láncok mindig egy, és csakis egy táblához tartozhatnak. Látni fogjuk, hogy néhány beépített lánc több mint egy táblához tartozik, de ez csakis a beépített láncokra áll. Egy másik táblából származó láncot nem keverhetünk bele egy felhasználó által készített láncba.

Az alap csomagszűrő tábla a `filter`, amely az `INPUT`, `FORWARD` és `OUTPUT` beépített láncokkal rendelkezik. A `filter` táblába kerülő felhasználói láncokhoz adott szabályok csak olyan célokra vonatkozhatnak, amelyek érvényesek az `INPUT`, `FORWARD` vagy `OUTPUT` láncokban. A `filter` táblán áthaladó csomagok az `INPUT`, `FORWARD` vagy `OUTPUT` láncok valamelyikén (és csakis azon az egyen) haladnak keresztül. Az `INPUT` lánc akkor kerül a képbe, ha a csomag célja a helyi rendszer. A `FORWARD` lánchoz akkor jut el a csomag, ha a helyi rendszeren keresztül egy másik rendszerhez továbbítódik. Végül az `OUTPUT` lánchoz egyedül olyan csomagok kerülhetnek, amelyek a helyi rendszerről származnak és külső célra irányulnak. Minden csomag csak egyetlen lánchoz kerülhet – szemben az IP Chains-megoldással, amely az `input` és `output`, illetve a `forward` láncot is használta, ha egy csomag keresztülment a rendszeren.

Figyeljük meg, hogy az előző bekezdésben az IP Tables láncok nevei nagybetűvel szerepelnek, míg az IP Chains láncnevek kisbetűsek. Szándékosan van így, és az írásmód változását hivatottak jelezni.

A `nat` tábla a hálózati cím átalakításáért felelős. Beépített láncai a `PREROUTING`, a `POSTROUTING` és az `OUTPUT`. Minden lánc egyetlen adott célt engedélyez. A `PREROUTING` a `DNAT` célt fogadja, a maradék láncok pedig az `SNAT` célt. Nemsokára kicsit bővebben is megismerkedünk velük.

A `mangle` (ferdítő) tábla az IP-címen kívül a fejlécben található adat eltorzítására szolgál: megjelölhetjük vele a csomagokat, megváltoztathatjuk a szolgáltatás típusát (TOS), vagy akár az életciklusértéket (time-to-live avagy `ttl`) is.

Szabályok

A szabálymeghatározó rész minden `iptables` parancs szíve. A szabály helyes felépítésével pontosan megadhatjuk, hogy mely csomagokra vonatkozzon. Ez a kiválasztó ismérv annyira általános vagy egyéni lehet, amennyire csak szeretnénk. A legtöbb esetben nem árt, ha egyéni meghatározásaink az általánosabbak előtt következnek.

Ebben a cikkben nem fogom túl sokáig boncolgatni a szabályokat, éppen csak hozzájuk adom a szükséges többszörös lehetőségeket (melyek közül néhánynak saját külön lehetőségei is vannak). Mindenkinek magának kell figyelnie arra, hogy a szabálynak értelme is legyen, például ne adjunk meg kimenő csatolófelületet egy `input` láncban, mivel a szabályhoz itt soha nem fog semmilyen találat tartozni. A parancs szerkezete lehetővé teszi, hogy lehetetlen szabályokat írjunk le, ami azonban semmiképpen sem túl jó ötlet. Ha kétségeink támadnak egy adott szabállyal kapcsolatban, célként adjuk meg a naplócélt, majd keltsünk olyan forgalmat, amely összeillik az adott szabállyal, hogy lássuk, valóban végrehajtódik-e. Ha szabályellenőrző eszközre van szükségünk, vessünk egy pillantást a SendIP-re (➲ http://www.earth.li/projectpurple/progs/sendip.html).

Netfilter engedélyezése a rendszermagban

A Netfilter beállítási menüpont eléréséhez előbb engedélyeznünk kell a *Code maturity-level* menüpontban a fejlesztés alatt álló, illetve befejezetlen meghajtókat (*development and/or incomplete code/drivers*). Ezt a kapcsolót beállítva lépjünk a *Networking menüpontra*, ahol be kell állítanunk a csomagszűrő szolgáltatást (*Network packet filtering (replaces ipchains*). Hacsak nincs néhány gigabájt felesleges helyünk, jobb, ha a *Packet-filtering Debugging* (Network csomagszűrő nyomkövetés) nem kapcsoljuk be.

Most lépjünk az *IP: Netfilter Configuration* (Netfilter-beállításokra), hogy elérjük a Netfilter-modulok almenüit, amelyekben az összes lehetséges pontot kiválaszthatjuk modulként. A modulok a legtöbb esetben csak akkor töltődnek be, ha szükség van rájuk. E sorok írásakor négy kivétel létezik, ezekkel külön foglalkoztunk.

Ha Ipv6-ot szeretnénk használni, ki kell választanunk az IPv6 protokollt, majd be kell lépnünk az *IPv6: Netfilter Configuration* menüpontba. Itt is, akárcsak az előbb, modulként az összes pontot kiválaszthatjuk.

Célok

A Netfilter négy beépített céllal rendelkezik: `ACCEPT` (elfogad), `DROP` (dob), `QUEUE` (sor) és `RETURN` (visszatérés). A `DROP` cél az `ipchains` `DENY` célját váltja fel. Az összes többi cél a célként betölthető modulokon alapul. Ide tartozik a `REJECT` (elutasít), a `LOG` (naplóz), a `MARK` (jelöl), a `MASQUERADE` (álcázás), a `MIRROR` (tükröz), a `REDIRECT` (átirányít) és a `TCPMSS`. A végcélok (terminal), mint például az `ACCEPT`, a `DROP`, a `REJECT`, a `MASQUERADE`, a `MIRROR` és a `REDIRECT`, mindig befejezik a láncot, a `LOG` viszont nem vet neki véget.

A `LOG` ugyanakkor nem is fogadja, utasítja vagy veti el a csomagot, így a lánc értelmezése folytatódhat. Ezért az

`ipchains -l` kapcsoló tulajdonképpen szintén egy cél-, de nem végtípusú. A lánc maradék része is végigfut, mígnem egy házirendszabályt talál.
A házirendszabály a teljes láncra vonatkozó, teljes körű szabály. Ha `FORWARD` láncunk egy `DROP` szabályt tartalmaz, és semmilyen más korábbi szabály nem jelzett találatot a láncban, a csomag ennél a szabálynál fog végezni. A házirendszabály csak valamelyik beépített cél lehet, a `REJECT` szabályt azonban nem adhatjuk meg házirendszabálynak.

Példák

Mielőtt belekezdenénk a példákba, állítsunk be néhány dolgot. A parancsfájlok hasznos dolgok, különösen amelyek az ***rc.local*** könyvtárban futnak (bárhol legyen is a rendszerünkön). Írjunk hát példánk részeként egy ***rc.iptables*** parancsfájlt, hogy már rendszerindításkor kapcsoljuk a Netfiltert (lásd az ***1. listát*** – megjegyzés: minden `iptables` szabály $IPT-vel kezdődik és sortörés nélkül kell folytatódnia a sor végéig, azaz a következő parancsig. Semmilyen szabályt sem szabad a sor közepén megtörni!)
Beállítottunk néhány indításhoz szükséges változót, leállítottuk a forgalom továbbküldését a rendszeren, majd beillesztettünk néhány modult. Az `ip_tables` modul teszi lehetővé a számunkra, hogy szabályokat kezdjünk el írni. Az `ip_nat_ftp` modul akkor szükséges, ha a `NAT` táblát használjuk (a később ismertetésre kerülő) `SNAT` vagy `MASQUERADE` szolgáltatáshoz, ugyanakkor működő FTP-t is szeretnénk. Az `ip_conntrack_ftp` az FTP-kapcsolat követését teszi lehetővé. Ez a modul önműködően betölti az `ip_conntrack` modult, mivel tőle függ. Ha nincs szükségünk az `ip_conntrack_ftp` modulra vagy nem akarjuk betölteni, de azt szeretnénk, hogy a tűzfal IP-töredezettségmentesítést végezzen (ami jó dolog), az `ip_conntrack_ftp`-t az `ip_conntrack`-kal helyettesíthetjük. Nézzük tovább a parancsfájlunkat:

```
for i in filter nat mangle
do
$IPT -t $i -F
$IPT -t $i -X
done
```

A fenti sorok minden szabályt kiürítenek a `-F` értékeként megkapott láncban. Mivel a `-F` értékként semmilyen láncnevet nem kapott meg, az összes láncot ki fogja üríteni. Figyeljük meg, hogy ez a ciklus miatt minden táblán végrehajtódik. Ha nem használunk egy adott táblát, kitörölhetjük a listából. Ahogy az egyes ciklusok végrehajtódnak, megfigyelhetjük, hogy új modulok töltődnek be: először az `iptables_filter` modul, majd az `iptables_nat` és végül az `iptables_mangle`. Ha a `mangle` kulcsszót eltávolítjuk a ciklusból, az `iptables_mangle` modul nem fog betöltődni. A nem használt modulokat tetszés szerint el lehet távolítani. Ip Chains használata esetén ugyanehhez a feladathoz valami ilyesmit kellett volna beírni: `ipchains -F -X`.
Mielőtt folytatnánk, tételezzük fel a következő felállást: van egy otthoni felhasználó és három rendszer, amely az Internetet a mi tűzfalként működő PC-nken keresztül éri el. Az elérés betárcsázás-alapú (ppp0). Ha tényleges beállításunk ettől eltérő, helyettesítsük a megfelelő külső csatolóeszközt a ppp0 helyére. A rendszerek saját hálózatukon kívül semmilyen az eth0-n található szolgáltatást nem ajánlanak fel, ugyanakkor azt szeretnénk, ha az összes belső rendszer képes lenne szörfözni a Világhálón. Ehhez az első példához tételezzük fel, hogy az ISP-től kapott dinamikus IP-címmel rendelkezünk (lásd a 2. listát a 20. CD-n).

Mivel tűzfal-számítógépünk egyben munkaállomás is, saját forgalmát is maga szabályozza. Bár tűzfal esetében nem túl jó ötlet (az ilyen feladatokat többnyire külön rendszerek végzik), egy otthoni hálózatban azért nem igazán van szükségünk külön? tűzfalra. Ezt figyelembe véve emlékezzünk rá, hogy az IP Tables és IP Chains közti egyik különbség az `INPUT`, `FORWARD` és `OUTPUT` láncok eltérő kezelése. Az IP Chains használatakor a `FORWARD`-ra kerülő csomagok az `INPUT`-ról érkeznek, és miután keresztülhaladtak a `FORWARD`-on, az `OUTPUT`-ra utaznak, ezért a szabályainkat az `INPUT` láncban elhelyezve a `FORWARD` lánc csomagjait is máris biztonságban tudhatjuk. Az IP Tables megvalósítás az `INPUT`-ot ellenben csak a helyi rendszerhez használja, a `FORWARD`-ot pedig a többi rendszerhez. A mi esetünkben mind a `FORWARD`, mind az `INPUT` láncokban azonos szabályokra lesz szükségünk. Hogy a szabályokat ne kétszer kelljen leírnunk, készítsünk egy `tcprules` nevű felhasználói láncot és az `INPUT` és a `FORWARD` láncból egyaránt hívjuk meg. Parancsfájlunkat így folytassuk:

```
$IPT -t filter -N tcprules
```

A szabály IP Chains-megfelelője ugyanez lenne, kivéve a `-t` szűrőkapcsolót:

```
ipchains -N tcprules
```

Most lássunk egy kis Netfilter-varázslást. Meg szeretnénk akadályozni, hogy a rendszerünkhöz mások kívülről hozzákapcsolódjanak, de engedélyezni szeretnénk saját felhasználóink kifelé irányuló kapcsolatait. A következő szabályok kihasználják a Netfilter kitűnő képességeit:

```
$IPT -t filter -A tcprules -i ppp+ -m state
➥--state ESTABLISHED,RELATED -j ACCEPT
```

```
$IPT -t filter -A tcprules -i ! ppp+ -m state
➥--state NEW -j ACCEPT
$IPT -t filter -A tcprules -i ppp+ -m state
➥--state NEW,INVALID -j DROP
```

Ha IP Chains alatt szeretnénk valami hasonlót elérni, ahhoz leginkább a syn csomagok elutasításával juthatunk el, a ppp+ csatolófelületen:

```
ipchains -A input -i ppp+ ! -y -j DENY
```

Ezen a ponton érdemes néhány gyors megjegyzést közbeszúrni: a "!" megfordítja (negálja), ami utána következik, így a `! ppp+` ugyanaz, mintha minden más csatolót meghatároztunk volna (a mi otthoni felhasználónk esetében ez a `lo` és az `eth0`). A `"+"` a ppp végén azt jelzi a Netfilternek, hogy az adott szabály az összes ppp-csatolófelületre érvényes.

Az `ESTABLISHED`, `RELATED`, `NEW` és `INVALID` értékek közül az `ESTABLISHED` engedélyezi a forgalom folytatását, ha korábban már mindkét irányban volt forgalom. Az `ESTABLISHED` nyilvánvalóan minden TCP-kapcsolatra értelmezhető, de UDP-kapcsolatra is vonatkozhat, például DNS-lekérdezésekre és traceroutes kérelmekre, vagy akár az ICMP pingre. Az első lépés annak kiderítésére, hogy a kapcsolat létezik-e már a kapcsolatkövetési táblában (/proc/net/ip_conntrack), a csomagok ellenőrzése. Ha igen, a láncok nem futnak le, az eredeti szabály lesz érvényes, így a csomag áthaladhat. Néhány esetben a Netfilter akár gyorsabb is lehet, mint az elődei, hála ennek az ellenőrzésnek. A `RELATED` érték számos dolgot takar: működő FTP-re lehet alkalmazni, amely a megfelelő kapcsolatokat a húszas kapun hozza létre, de a TCP-kapcsolathoz tartozó ICMP-forgalomra is alkalmazható. A `NEW` érték azokra a csomagokra vonatkozik, melyeknek csak a SYN bitjük van beállítva (és ACK bitjük nem működik). Az `INVALID` azokra a csomagokra vonatkozik, amelyek az XMAS fa-keresés (XMAS tree scan) szerint hibás beállításokkal rendelkeznek.

A fenti szabályok lehetővé teszik, hogy a belső rendszerek belsőleg és külsőleg is átjuttathassák az új kapcsolatokat, ugyanakkor ne engedélyezzék a kívülről jövő vagy hibás csomagokat.

Mivel a hálózatunkat álcázás (masquerading) segítségével szeretnénk a tűzfal mögé tenni, be kell állítanunk pár álcázási szabályt. Ez sokban hasonlít az IP Cainsben használt folyamathoz. Feltételezve, hogy belső hálózatunk címe 192.168.0.0/24, a következő szabályt kell használnunk:

```
$IPT -t nat -A POSTROUTING -o ppp+
➥-s 192.168.0.0/24 -d 0/0 -j MASQUERADE
```

Az egyetlen különbség, amit az IP Chainst használóknak feltűnhet, hogy `-o ppp+` utasítást használtunk `-i ppp+` helyett. Ez azért van, mert míg IP Chains alatt a csatolófelületekre a `-i` segítségével hivatkozhatunk, IP Tables alatt a `-i` a bemeneti (input) csatolófelületet jelenti, a `-o` pedig a kimeneti csatolófelület jele. Ha a fenti sorba véletlenül `"-i ppp+"`-t írunk, az álcázás nem fog elindulni, hiszen a szabály soha nem fog semmivel sem egyezni.

Fejezzük be a parancsfájlunkat, és a fenti TCP-szabályokat szükség szerint illesszük be, hogy beindíthassuk a szűrőtábla-rendszabályokat és újraindítsuk az `ip_forwarding` szolgáltatást:

```
$IPT -t filter -A INPUT -j tcprules
$IPT -t filter -A FORWARD -j tcprules
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Alapértelmezés szerint a szűrőtábla szabályok `ACCEPT`-értékűek. A Netfilter elegáns megoldásai miatt azonban – az IP Chainsszel ellentétben – így is megfelelő biztonságot nyújt. Ha visszagondolunk az eddig alkalmazott szabályokra, láthatjuk, hogy az alapértelmezett rendszabály semmiféle ártalmat nem okoz; sőt, eszerint tulajdonképpen semmi sem érkezhet meg. Néhányan azonban úgy vélekednek, hogy jobb mindinkább a biztonságra törekedni, így például dobhatunk mindent, ami nincs már korábban megcímezve. Figyeljük meg, hogy a rendszabályoknak nincsen céljuk, tehát nem is használhatjuk a `-j` kapcsolót. Ez az, amiért rendszabályok csak beépített célok lehetnek. A rendszabályok nem igazi célok. Ha tényleg `REJECT`-típusú rendszabályt szeretnénk, valami ilyesmit kell beírnunk:

1. lista Az „rc.iptables” parancsfájl

```
#!/bin/sh
# elsıkØnt adjuk meg a megb zhat   svØnyeket
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/
usr/bin:/usr/local/bin

export PATH

# m dos tsuk a k vetkezıket, hogy az
# IP Tables futtathat  ÆllomÆnyra mutassanak:
IPT=/usr/local/sbin/iptables

echo 0 > /proc/sys/net/ipv4/ip_forward

# nØhÆny modul, amit be akarunk t lteni
modprobe ip_tables
modprobe ip_nat_ftp
modprobe ip_conntrack_ftp
```

```
$IPT -t filter -A tcprules -i ppp+  -j REJECT
➥--reject-with icmp-host-unreachable
```

A fenti szabály a ppp csatolófelületen bejövő összes csomagra érvényes lesz. Ügyeljünk rá, hogy utolsóként szerepeljen, különben semmi sem fog átjutni ezen a szabályon.

Bár kiadhattunk akár egy

```
$IPT -f filter -A tcprules -j REJECT
➥--reject-with icmp-host-unreachable
```

szabályt is, azt tapasztaljuk, hogy a saját belső gépeink is blokkolva lesznek, mivel ez a szabály az összes csatolófelületre vonatkozik (ezt a rendszabályok használatakor is figyelembe kell venni).

Ezen a ponton megemlíteném, hogy az IP Chainsszel ellentétben a megfelelő cél megtalálása nem feltétlenül jelenti a lánc végét is. Ha például a `tcprules`-ban a következő szabályt használjuk:

```
$IPT -t filter -I INPUT -p ICMP -m icmp
➥--icmp-type echo-request -m limit
➥--limit 1/minute -j LOG --log-prefix
➥"ICMP-packet "
```

akkor a következő szabály (amely a csomaggal vagy egyezik, vagy nem) szintén végrehajtódik. Amennyiben a szabály nem ejti, utasítja, fogadja el vagy sorolja be (QUEUE) a csomagot és nem visszatérés (RETURN), akkor a lánc folytatódni fog.

Az IP Chains-használók figyelmét felhívnám rá, hogy IP Tables alatt a `LOG` önmaga is cél, és nem egyszerűen csak egy `-l` kapcsoló a cél mögött, mint az IP Chainsben. Ez bizonyos rugalmasságot nyújt, amint az a fenti szabályból is látható. Az egyezés korlátozása megakadályozza, hogy valaki rossz szándékkal

4. lista DNS-bejegyzés a levelezőkiszolgálóhoz

```
$IPT -t filter -N tcprules
$IPT -A tcprules  -i eth1 -m state
➥--state ESTABLISHED,RELATED -j ACCEPT
$IPT -A tcprules -i ! eth1 -m state
➥--state NEW -j ACCEPT
$IPT -A tcprules -i eth1 -p tcp
➥--dport 25 -m state --state NEW -j ACCEPT
$IPT -A tcprules -i eth1 -m state
➥--state NEW,INVALID -j DROP
$IPT -t nat -A PREROUTING -d 209.127.112.17
➥-p tcp --dport 25 -j DNAT
➥--to-destination
➥192.168.0.2:25
$IPT -t nat -A POSTROUTING
➥-o eth1 -s 192.168.0.0/24 -j SNAT
➥--to-source 209.127.112.17
$IPT -A INPUT -j tcprules
$IPT -A FORWARD -j tcprules
echo 1 > /proc/sys/net/ipv4/ip_forward
```

túlterhelje a naplónkat. A `LOG` cél az üzeneteihez előtagot is fűzhet, így a `grep` paranccsal a naplókból könnyen kikereshetők. Minden IP Chains-felhasználó jól tudja, hogy a láncokat a következőképpen lehet megtekinteni:

```
ipchains -L -n
```

Netfilter használatakor azonban a láncokat tábláról táblára kell megnéznünk:

```
iptables -t filter -L -n
iptables -t nat -L -n
iptables -t mangle -L -n
```

Használhatjuk a `-v` kapcsolót is, hogy az egyes szabályokról részletesebb ismeretekhez jussunk:

```
iptables -t filter -L -nv
```

Következzék egy példa, amely az új `SNAT`- és `DNAT`-képességeket mutatja be. Ha valaki érti a `mangle` táblát és az is világos számára, hogy mire való, annak valószínűleg nemigen van szüksége cikkünk elolvasására, és hamarosan elküldi nekem azokat a hibákat, amelyeket észrevett.

Ebben a példában azt fogjuk feltételezni, hogy otthoni felhasználónk széles sávú eléréssel rendelkezik, és az `eth0`-t használja a belső hálózathoz, illetve az `eth1`-et a külsőhöz. A parancsfájl az előzőekhez hasonlóan indul, de amikor a `tcprules` szabályokhoz érünk, egy kicsit mást fogunk alkalmazni. Jelen esetben tételezzük fel, hogy a felhasználó a 209.127.112.17 számú állandó IP-vel rendelkezik, ami egy kicsit megváltoztatja a dolgokat (lásd a 3. listát a 20. CD-n). Azt is elképzelhetjük, hogy a felhasználó saját tartománynévvel rendelkezik, és saját levelezőkiszolgálót futtat a 25-ös kapun, a belső rendszeren található a 192.168.0.2 IP-számon (a tűzfal száma 192.168.0.1). A DNS bejegyzése a 209.127.112.17 címet jelöli meg levelezőkiszolgálóként, ahogyan azt a ***4. listában*** láthatjuk.

Az első két `tcprules` szabály ugyanaz, mint a fenti példában. Mielőtt azonban minden más kapcsolatot elvetnénk, elfogadjuk a 25-ös kapun keresztül érkező csomagokat. Ezután a `nat` táblában elfogjuk a 25-ös kapcsolatot és átirányítjuk egy másik belső gép ugyanilyen számú kapujára. Ezt egyébként az összes kapcsolatunkkal megtehetjük. Ne feledjük viszont, hogyha a DNS-t is ilyen módon irányítjuk át, akkor UDP- és TCP-átirányításra egyaránt szükségünk lesz. Általában – hacsak valaki odakintről nem akar zónaátirányítást a TCP- részeket nyugodtan elvethetjük, és csak az UDP-részt engedjük át.

Azt is figyeljük meg, hogy a `MASQUERADE` használata helyett a kimenő kapcsolatokhoz a `SNAT`-ot használjuk célként. Ha valaki csodálkozna, az `S` a forrást (source) jelenti, ezt fogjuk megváltoztatni. A `DNAT` esetében a csomag célját változtatjuk meg. A `--to-source` érték IP-címtartományok fogadására is képes, így kívülről a tűzfal több gépnek is látszódhat. Ha például az ISP-től öt felhasználható IP-számot kaptunk, a kimenő kapcsolatokhoz mind az ötöt felhasználhatjuk. A különféle szolgáltatásokat külön IP-számokhoz rendelhetjük, így akár öt rendszer is válaszolhat DNS-lekérdezésekre, tárolhat weblapokat, fogadhat leveleket stb. A Netfilter segítségével kezdetleges terheléskiegyenlítést is készíthetünk. Ha céltartományt használunk a `DNAT`-hoz, a legkevesebb kapcsolattal rendelkező rendszer (ami nem feltétlenül jelenti a legkevésbé terhelt rendszert) kapja az új kapcsolatot.

Amit még érdemes tudnunk, hogy elindulhassunk, az, hogy miképpen növelhetjük meg az egyszerre követett kapcsolatok számát. Ez a szám alapértelmezés szerint a rendszeren rendelkezésre álló memória függvénye. A szám azonban megnövelhető. A következőképpen találhatjuk meg:

```
cat /proc/sys/net/ipv4/ip_conntrack_max
```

Az én 256 MB memóriájú rendszeremen ez a szám 16376.

Befejező megjegyzések

A Netfilter használatával minden erőlködés nélkül növelhetjük otthoni rendszerünk biztonságát: a kapcsolatkövetés józan használatával. Számos más lehetőség is rendelkezésünkre áll – ez a cikk épp csak a felszínt kapargatta meg. Mindenkinek ajánlom Rusty (korábban már említett) „Unreliable Guides” című írását, mely a Netfilter honlapon érhető el.

Egyszerű igényekkel rendelkező otthoni felhasználók jobb, ha egyszerű tűzfalat építenek, ezért nem javaslok sok elérhető tűzfaleszközt és parancsfájlt, hiszen mindössze felesleges bonyolultságot visznek a tűzfalunkba. Ha nem értünk meg egy szabályt, ne építsük be. Az első három meghatározó szabály (a `-m state` szabály alkalmazásával) már meglehetősen jó alapot ad. Amennyiben azonban a támadó már korábban bejutott a rendszerbe, a szabályok nem sokat érnek. Nem védenek meg továbbá a levélalapú trójaiaktól sem, megóvnak viszont a közvetlen támadásoktól. Azt javaslom, ha nem használunk IRC-t, naplózzuk és vessük el az IRC-kapcsolatokat:

```
$IPT -j filter -I tcprules -p tcp
➥--destination-port 6667 -j LOG
➥--log-prefix "IRC attempt "
$IPT -j filter -I tcprules 2 -p tcp
➥--destination-port 6667 -j DROP
```

Továbbá ha nincs rá szükségünk, hogy bárki is belépjen a rendszerünkbe, semmilyen kaput se nyissunk meg (ahogyan azt a második példában láthattuk). Cikkünk nem arról szólt, hogyan állítsuk be helyesen a hálózatot az Internet által elérhető rendszerek és a megbízható belső rendszerek elkülönítésére. Ha ilyen összetett rendszerre van szükséged, ideje, hogy egy hozzáértő segítségét kérd!

David A. Bandel
(dbandel@pananix.com) jelenleg Panamában él, Linux- és Unix-tanácsadással foglalkozik. Társszerzője a „Que Special Edition: Using Caldera OpenLinux” című könyvnek.