Tűzfal kialakítása Zorp segítségével

Tűzfalszolgáltatásokat minden fő rendszermagváltozat nyújt a 2.0 óta, bár ez volt a rendszermag egyik legváltozékonyabb része. Tűzfalunk kialakításakor folyamatosan egyre több lehetőség áll a rendelkezésünkre, ám egyvalami semmi nem változott: az átvitt, illetve megszűrt forgalomra minden rendszermag csomagok sorozataként tekint.

Az egyszerű csomagszűrők (Linux 2.0: IP Fwadm, 2.2: IP Chains), illetve az állapottartó csomagszűrők (Linux 2.4: IP Tables) jellemzője, hogy elsősorban a csomagok fejlécét vizsgálják, míg az adatrésszel nem, vagy csak kis mértékben foglalkoznak.

Hogy megértsük, mit is jelent ez pontosan, tegyünk egy kis kitérőt: az IP-alapú hálózatok egy ötrétegű hálózati modellnek feleltethetők meg (szemben az ISO OSI által leírt hét réteggel), a rétegek a következők: fizikai, adatkapcsolati, hálózati, szállítási és alkalmazási. Az alsó három réteg együtt nyújtja az IP-szolgáltatást, a szállítási réteg felel a TCP-ért, illetve az UDP-ért, az alkalmazási rétegben pedig az adott feladatnak megfelelő protokoll kap helyet.

Alkalmazási protokoll lehet például a levelek letöltésére való POP3 vagy a webezéshez használt HTTP, de minden feladathoz különböző protokoll tervezhető. Az alkalmazási protokollok között léteznek nyílt szabványúak, amelyek jól meghatározottak (HTTP, FTP), illetve teljesen zárt, ismeretlen protokollok is (Oracle SQL*Net).

A csomagszűrők az IP-verem alsó négy rétegével foglalkoznak, az alkalmazási protokollal (tehát a TCP-csomagok adatrészével) egyáltalán nem. Tekintve, hogy az alsó négy réteg fejlécei a csomag méretének körülbelül három százalékát jelentik (ethernet esetén), az ellenőrizetlen adatmennyiségek száma meglehetősen nagy.

Az IP-protokollverem alsó négy rétegének megvalósítását (a fizikaitól a szállítási rétegig) általában az operációs rendszer tartalmazza. Az alkalmazási réteget különböző, az operációs rendszerre épülő programok valósítják meg. Az alsóbb rétegek megvalósítása alaposan kipróbált, működése megbízható, hiszen minden hálózati program épít rájuk. Az alkalmazások minősége viszont gyakran elmarad ettől, több-kevesebb hibát tartalmaznak, ami sok esetben biztonsági sérülékenységhez vezethet.

A betörők a hálózaton keresztül a programok biztonsági réseit próbálják meg kihasználni, és mivel az egyetlen kapcsolat a programmal maga az alkalmazás protokollja, a biztonsági hibák kihasználása gyakran megjelenik az alkalmazási protokollban – ha például túl hosszú felhasználónevet küldünk egy kiszolgálónak, tömbtúlcsordulás következhet be.

Csomagszűrésnél jóval nagyobb biztonságot érhetünk el, ha a csomagok fejrésze mellett az alkalmazási protokollt is elemezzük, mely a csomagok adatrészében helyezkedik el (a TCP-, illetve az UDP-adatrészében).

Az ellenőrzést az alkalmazástól függetlenül egy megbízható elem végzi, amely a nem kívánatos részeket képes visszautasítani.

Ilyen ellenőrzést képesek megvalósítani az úgynevezett alkalmazásszintű átjárók, avagy a proxytűzfalak. Ilyen program például a Gauntlet, az TIS fwtk, a T.REX, illetve a Zorp is.

Telepítés

A Zorp fejlesztése Debian-rendszeren történik, ezért ezen a Linux-változaton lesz a legkönnyebb dolgunk, de természetesen más terjesztések is alkalmasak a futtatására.

Itt jegyezném meg, hogy a tűzfalakra fokozottabban áll, hogy ne futtassunk rajtuk felesleges szolgáltatásokat, és minden szempontból tegyük őket biztonságosabbá (setuid bitek eltávolítása, programok `chroot`-tal történő futtatása stb.).

A Zorp jelenleg a 2.2-es rendszermaggal működik együtt a legjobban, bár megszabott szolgáltatásokkal a 2.4-es rendszermaggal is használható (transzparencia csak egycsatornás TCP-alapú protokollokkal érhető el).

A rendszermagon kívül a következő csomagokra lesz szükségünk:

- openssl 0.9.6, glib 1.3.1 (pontosan az 1.3.1 szükséges, mivel a változtatások miatt a későbbi glib-változatokkal nem működik együtt),
- libcap 1.10, python 1.5.2 (figyeljünk rá, hogy legyen benne száltámogatás(thread)),
- python-extclass 1.2 (ExtensionClass néven is ismert és mostanában a Zope részeként terjesztik).

Ha a fenti csomagokat és a hozzájuk tartozó fejlesztői csomagokat feltelepítettük, neki is állhatunk a Zorp fordításának:

```
# tar xfz zorp-1.4.0rc17.tar.gz
# cd zorp-1.4.0rc17
# ./configure && make && make install
```

Ha a fordítás sikeresen lefutott, a Zorp programfájljait a */usr/local* könyvtárban találjuk meg. Figyeljünk, hogy a */usr/local/lib* könyvtár szerepeljen a dinamikus szerkesztő beállításában (*/etc/ld.so.conf* fájl), majd futtassuk le az `ldconfig` parancsot.

Beállítás

A Zorp az átmenő adatfolyam alkalmazásszintű elemzéséért felelős, de természetesen csomagszintű szűrést is alkalmazunk, melyet a rendszermag által nyújtott csomagszűrő szolgáltatásokkal érünk el. A csomagszűrés célja kettős:

1. megtiltjuk az átmenő forgalmat, és gondoskodunk róla, hogy a csomagtovábbítás helyett a proxykhoz kerüljön,
2. szabályozzuk a tűzfalat elérő csomagokat.

Egy Zorp-alapú tűzfal beállítása legalább a csomagszűrő-, valamint a proxyk beállításából tevődik össze (a valóságban természetesen további programokra is szükségünk lehet).

Az IP Chains segítségével alapértelmezésként mindent tiltunk, az engedélyezett forgalmat pedig egy `REDIRECT`-szabály segítségével a proxyhoz irányítjuk.

Amennyiben az ábrán látható hálózati felépítéssel rendelkezünk, a következő szabály „fogja meg” és téríti el a belső hálózatról kifelé irányuló webes forgalmunkat:

```
# ipchains -A input -i eth0
# -p tcp -d 0/0 80 -j REDIRECT 50080
```

A fenti szabály feltételezi, hogy a tűzfalgép 50 080 kapu-

ján egy olyan proxy hallgatózik, amely a forgalom közvetítéséről gondoskodik. Figyelem, a `REDIRECT`-szabályok működéséhez az IP-csomagok továbbítását a */proc/sys/net/ipv4/ip_forward* fájlban be kell kapcsolnunk, ezért gondoskodnunk kell róla, hogy ennek ellenére valós továbbítás a következő szabállyal ne történjen:

```
# ipchains -A forward -j DENY -l
```

Az 1. listán (http://www.linuxvilag.hu/Zorp/1.html) egy teljes csomagszűrő-beállítás látható – az `ipchains-restore` programnak megfelelő formátumban.

A Zorp két állományból áll, amelyet, ha a programot a fenti módon telepítettük, a */usr/local/etc/zorp* könyvtárban találhatunk, csomagból való telepítés esetén pedig a */etc/zorp*-ban.

Az egyik állomány az *instances.conf*, amely azt írja le, hogy a tűzfalon milyen és hány Zorp-példány fusson. Érdemes minden zónának kijelölt példányt létrehozni, mely majd az adott zónából származó kéréseket szolgálja ki. Az *instances.conf* minden sora egy-egy példányt határoz meg: az első szó a példány nevét, a sor további része pedig a Zorp programnak átadandó parancssori kapcsolókat adja meg. A mintaállományt a 2. lista (http://www.linuxvilag.hu/Zorp/2.html)) tartalmazza.

A létrehozott példányokat a `zorpctl` parancs segítségével indíthatjuk el, illetve állíthatjuk le. A másik beállítási fájl az alkalmazandó szabályrendszert tartalmazza, neve az *instances.conf*-ban egy kapcsolóval megadható. A Zorp esetében a szabályrendszert Python segítségével írjuk le, így ez a fájl egy Python-modul lesz, alapértelmezett helye pedig a *$prefix/etc/zorp/policy.py*. Mivel a fájl Python-modul, egy dolgot szem előtt kell tartanunk: a blokkok kezdetét és hosszát beljebbkezdés (indent) jelzi. A beljebbkezdésre vonatkozó egyetlen megkötés az, hogy egy blokkon belül azonos méretű kell legyen. Szóközt és tabulátort egyaránt használhatunk, az utóbbi mérete viszont az értelmező szerint 8 szóköz, ha tehát tabulátort használunk, a szövegszerkesztőnkben is ezt az értéket állítsuk be. Ha nem ügyelünk erre a szabályra, szintaktikai hibát kapunk.

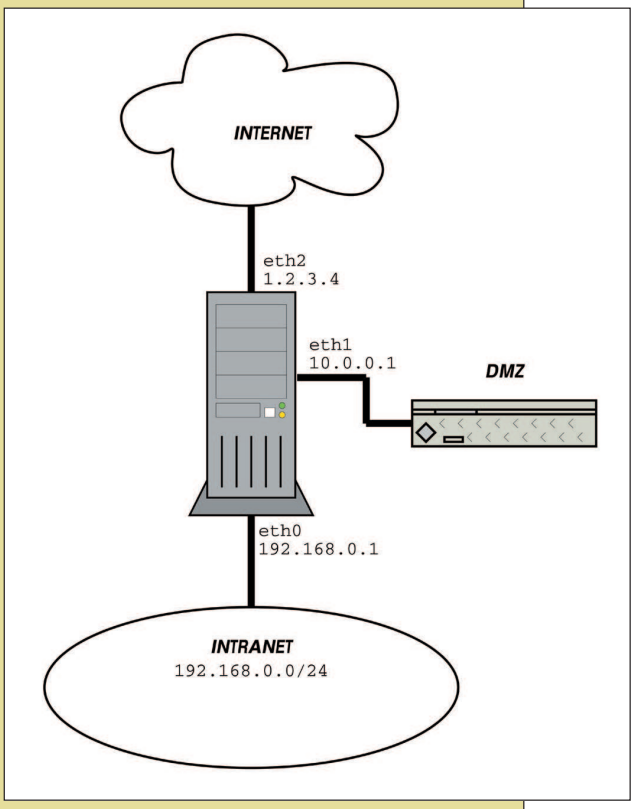
Nézzük, hogyan is épül fel egy beállítási fájl:

1. importálási rész, ami a Zorp által nyújtott szolgáltatásokat teszi elérhetővé,
2. zónameghatározási rész, mely a Zorp számára a környező hálózat felépítését írja le,
3. a proxy beállítása,
4. példánymeghatározások

Az egyes részeket a 3. listán (http://www.linuxvilag.hu/Zorp/3.html) láthatjuk.

Egy szolgáltatást a következő módon vehetünk fel: kiválasztjuk a használni kívánt Zorp-proxyt, majd eldöntjük, hogy a saját igényeink szerint kívánjuk-e testreszabni. A Zorpban minden proxynak, illetve a kapcsolódó beállításoknak egy-egy Python-osztály feleltethető meg.

A testreszabás egy kész osztályból való származtatást, illetve a megfelelő értékek (tulajdonságok – attributes) beállítását jelentik. Az egyszerűség kedvéért új szolgáltatásunk legyen egy átlátszó (transparent) webszolgáltatás a belső hálózatról az Internetre, melyet a Zorp alap HttpProxy-osztályával továbbítunk. Adjunk nevet ennek a szolgáltatásnak, legyen a továbbiakban „intra_HTTP”. A példányleíró részben minden Zorp-példányt egy Python-függvény jelképez, amelynek neve megegyezik a Zorp-példány nevével. Ez a függvény felelős az adott



példány betöltéséért. Új szolgáltatásunk létrehozásáért a példánynak megfelelő függvénybe vegyük fel a következő két sort:

```
Service("intra_HTTP", HttpProxy)
Listener(SockAddrInet
➥("192.168.1.1", 50080),
➥"intra_HTTP")
```

A fenti két sor létrehoz egy intra_HTTP nevű szolgáltatást, majd egy hallgatózó foglalathoz (socket) köti, ami az 50 080-as kapun várja a kéréseket.

Ezekután a szolgáltatást engedélyeznünk kell: a beállítófájlban a zónameghatározásoknál a belső hálózatot leíró részhez az engedélyezett kifelé tartó szolgáltatásokhoz (outbound_services tömb) adjuk hozzá. Ehhez hasonlóan a szolgáltatás céljaként szereplő zónáknál ugyanezt az inbound_services halmazba is vegyük fel. Láthatjuk, hogy a Zorp ugyanazon a kapun hallgatózik, ahová az előző példában szereplő IP Chains-szabály a kimenő http-forgalmat eltérítette. Tehát a teljes kimenő http-forgalom a Zorp http-proxyn keresztül megy át, mely az adatfolyamot alkalmazásszinten elemzi.

Szolgáltatás létrehozásakor további lehetőségeink is adódnak, például eldönthetjük, hogy a célkiszolgáló címe