Önműködő tűzfal naplókövetéssel

Néhány módszert és parancsfájlt mutatunk be a szűrők által készített naplófájlok önműködő nyomon követéséhez.

A tűzfalak olyan számítógépek, amelyek legfőbb feladata két hálózat közti hálózati forgalom egyes elemeinek szűrése. Általában azért alkalmazzák őket, hogy a belső hálózatot (LAN) megóvják az Internet egyéb részeitől. A belső háló összes gépének egyedi védelme sokkal költségesebb és időigényesebb lenne, mint egyetlen tűzfalat telepíteni, karbantartani és figyelemmel kísérni. A tűzfal különösen azoknak az intézményeknek nélkülözhetetlen, amelyek folyamatosan csatlakoznak az Internethez. A hálózati beállítástól függően az útválasztót (router) is be lehet állítani csomagszűrési feladatra, általában azonban sokkal kényelmesebb egy külön gépet működtetni tűzfalként. Mivel hihetetlen biztonságossá tehetők és az áruk is igen kedvező, Linux-alapú gépekből nagyon hatékony tűzfalak építhetők.
A 2.2.x változatú Linux-rendszermag esetében a tűzfal telepítése az IP Chains segítségével történik, míg az új, 2.4.x rendszermagban ezt a feladatot az IP Tables tölti be. Egy valódi tűzfal összeállításának leírása meghaladná cikkünk kereteit; az érdeklődő olvasók további adatokat az ipchains HOWTO-ban (2.2.x magokra vonatkozóan) és a Linuxvilág honlapján (➲ http://www.linuxvilag.hu/static.phtml?file=hogyanok/ipchains/ipchains-hogyan-00), illetve *Paul „Rusty" Russell* Packet-Filtering HOWTO-jában (2.4.x magokhoz) találhatják meg. Mindkettő – bármely kereső segítségével – fellelhető az Interneten.
A tűzfal felépítése azonban még kevés, ha igazán biztonságos rendszert szeretnénk, a tűzfalat felügyelni is kell. Ebben a cikkben azt mutatjuk be, miképpen lehet felépíteni és használni az inside-control elnevezésű webalapú IP Chains figyelőrendszert.
A tűzfalfigyelő rendszerek két fő feladattal bírnak: ellenőrizniük kell, hogy rossz szándékú betörő nem próbál-e bajt keverni a belső hálón, valamint hogy a LAN-on belüli felhasználók nem élnek-e vissza valamelyik internetszolgáltatással.

Tűzfalbeállítási példa

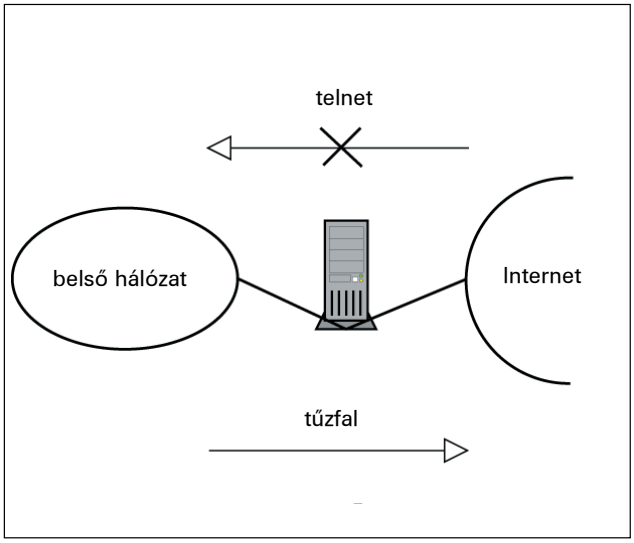
Lássunk egy nagyon egyszerű tűzfalbeállítást, amelyre ebben a cikkben a későbbiekben példaként fogunk hivatkozni.
Tegyük fel, hogy van egy 10.0.1.0/255.255.255.0 címtartományú belső hálózatunk; a linuxos átjáró, illetve tűzfal a 10.0.1.1 című csatolón keresztül éri el a belső hálót, és a 10.200.200.1 címen csatlakozik az Internethez (valójában egyik IP-cím sem nyilvános, így ez természetesen kitaláció). A tűzfal elindításának első lépése a két hálózati csatoló közötti átjáró felépítése:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

A következőkben a naplózó tűzfalat építhetjük fel az IP Chains segítségével. Előbb kiürítjük az összes korábbi szabályt, majd engedélyezzük a hurokeszköz (loopback interface) csomagjait és az összes ICMP-csomagot:

```
ipchains -F
ipchains -A input -i lo -j ACCEPT
ipchains -A input -p ICMP -j ACCEPT
```

Ezt követően megakadályozzuk, illetve naplózzuk az Internetről a belső hálózat felé irányuló Telnet-protokollkérelmeket:



A példatűzfal beállítása

```
ipchains -A input -p TCP -s 0.0.0.0/0
       ➥-d 10.0.1.0/24 23 -l -j DENY
```

Ugyanakkor engedélyezzük és naplózzuk a HTTP-protokollt, amely a belső hálózatról az Internetre irányul:

```
ipchains -A input -p TCP -s 10.0.1.0/24
       ➥-d 0.0.0.0/0 80 -l -j ACCEPT
```

Befejezésül beállítjuk a megengedő szabályokat (permissive policies):

```
ipchains -P input ACCEPT
```

Tűzfalunk megakadályoz és naplóz minden bejövő Telnet-kapcsolatot, engedélyez és naplóz minden kimenő HTTP-kapcsolatot, illetve engedélyez minden mást (lásd a *fenti ábrát*). Egy ilyen beállítás természetesen túl engedékeny ahhoz, hogy komoly védelmet nyújtson, de jól szemlélteti, hogy mire képes az önműködő naplóvizsgáló parancsfájl.
A tűzfal a kimenetét általában a /var/log/syslog vagy a /var/log/messages fájlok valamelyikébe küldi. A következő paranccsal kideríthetjük, hogy a kettő közül melyik az igazi:

```
grep -q "Packet log" /var/log/syslog && echo yes
```

Ha a kimenet *yes*, akkor a /var/log/syslog a keresett fájl, ha a kimenet üres, akkor valószínűleg a /var/log/messages lesz az. Ezt is ellenőrizhetjük a

```
grep -q "Packet log" /var/log/messages && echo yes
```

paranccsal.

1. lista 2.4.x IP Tables parancssorozat

```
iptables -F
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p icmp -j ACCEPT
iptables -A INPUT -p tcp -d 10.0.1.0/24 \
         --dport 23 -j LOG \
         --log-prefix "Packet log: " \
         --log-level info
iptables -A INPUT -p tcp -d 10.0.1.0/24 \
         --dport 23 -j DROP
iptables -A INPUT -p tcp -s 10.0.1.0/24 \
         --dport 80 -j LOG \
         --log-prefix "Packet log: " \
         --log-level info
iptables -A INPUT -p tcp -s 10.0.1.0/24 \
         --dport 80 -j ACCEPT
iptables -P INPUT ACCEPT
```

Ha mindkét parancs üres kimenetet ad, akkor a tűzfal nem működik, vagy még semmilyen naplózott forgalom (a példánkban Telnet vagy HTTP) sem haladt át a tűzfalon.

2.4.x magok és az IP Tables

2.4.x-es rendszermag és IP Tables esetén a dolog egy árnyalatnyit bonyolultabb. Először is: nem szabad elfelejtenünk a magba az összes csomagszűrő szolgáltatást belefordítani, ideértve a naplózás (LOG) részt is. Másodszor cseréljük az IP Chainst IP Tablesre, majd a láncok neveit nagybetűsre (pl.: az inputból INPUT lesz). Ezt követően változtassuk meg a célpontok neveit (DENY DROP-ra). Végül a kapuszámokat is másféleképpen kell megadni. Az *1. lista* a fentebb bemutatott 2.2.x parancsok 2.4.x-es változatát mutatja be.

IP Chains naplóformátum

Most vizsgáljuk meg tűzfalunk /var/log/syslog bejegyzéseit:

```
Jun 12 16:15:54 myfirewall kernel:
       ➥Packet log: input DENY eth1 PROTO=6
       ➥212.65.214.2:34251 10.0.1.2:23
       ➥L=52 S=0x10 I=24016 F=0x4000 T=53 SYN (#38)
```

Ezek szerint június 12-én, délután negyed ötkor a tűzfal (mely a nem túl ötletes myfirewall nevet viseli) elutasított és naplózott egy az eth1 hálózati csatolóra érkező (azaz az Internetről jövő) TCP-protokollú csomagot, mely a 212.65.214.2 címről (és a 34251 kapuról) indult, a 10.0.1.2 címre (a 23-as, azaz a Telnet-kapura) irányult, és 52 bájt hosszúságú volt. A többi részletet nyugodtan átugorhatjuk, kivéve egyet: a SYN azt jelenti, hogy ez a csomag volt a kapcsolat első csomagja. A gyakorlatban ez az adat nagyon hasznos lehet, mivel ennek alapján lehet megkülönböztetni a már létező kapcsolatok csomagjait (amiket esetleg a belső hálózat kezdeményezett), és azokat a csomagokat, amelyek új kapcsolatot próbálnak létrehozni az Internetről a belső háló felé. Általában engedélyezzük a *reply* (válasz) csomagokat (azaz azokat amelyeknek nincsen SYN-jelük), viszont tiltjuk a SYN-csomagokat, mert ezek azt jelentik, hogy valaki odakint valamelyik belső hálózati géppel kapcsolatot szeretne felépíteni. Természetesen a tűzfal állapotát a naplófájlok megfelelő bejegyzéseinek végigböngészésével is meg lehet vizsgálni, de ez csak akkor célszerű, ha valaki csak néhány gyanús kinézetű csomagot naplóz. Például némelyik általam beállított tűzfalnál úgy döntöttem, hogy naplózni fogom az összes olyan csomagot, amely az Internetről a 31337-es számú kapu irányába megy a belső hálózat számítógépein, mert ez a BackOrifice alapértelmezés szerint használt kapuja. Ha viszont valaki a tűzfalról kimutatást is szeretne készíteni, valószínű, hogy a naplófájl mérete meghaladja a napi öt megabájtot. Ilyen esetekben a naplófájl kézi vizsgálata már nem járható út. Ekkor nyújthat segítséget az önműködő naplófigyelés.
Ha 2.4.x mag tűzfalnaplókat vizsgálunk, a formátum más lesz:

```
Jun 12 16:15:54 myfirewall kernel
   ➥Packet log: IN=eth1 OUT=
   ➥MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00
   ➥SRC=212.65.214.2 DST=10.0.1.2 LEN=52 TOS=0x10
   ➥PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=34251
   ➥DPT=23 WINDOW=11592 RES=0x00 SYN URGP=0*
```

A minket érdeklő mezők: az SRC (forrás IP-cím), a DST (cél IP-cím), a SPT (forráskapu), a DPT (célkapu), illetve a SYN-kapcsoló megléte vagy hiánya.

Az inside-control szerkezete

A naplókövető felépítéséhez Perlt fogok használni. Természetesen nem ez az egyetlen lehetséges választás, sőt, ha valaki a legnagyobb sebességre törekszik, akkor tulajdonképpen jobban teszi, ha valamilyen fordítóval működő (compiled) nyelvet választ. Amikor ugyanezt a parancsfájlt C++ nyelven újraírtam, körülbelül százszázalékos sebességnövekedést tapasztaltam. Az inside-control parancsállomány egy főértelmező ciklusból és egy HTML-adatmegjelenítő ciklusból áll. Mivel ez CGI parancsfájl, olyan webkiszolgálón kell elhelyezni, amelyet felkészítettek CGI programok futtatására.
A kód, amint majd alább kiderül, az érthetőség érdekében feláldoz valamennyit a használhatóságból, továbbá kihagy pár hasznos részletet, például a hibaellenőrzést is. Többek között azt sem ellenőrzi, hogy a fájlmegnyitás sikeres volt-e, mielőtt olvasna abból.
Azt is érdemes megemlíteni, hogy a lenti kód 2.2.x-es magok csomagnaplózási formátumához lett igazítva. A 2.4.x változathoz illesztés a korábbiakban már megismert csomagnapló-példa alapján magától értetődik.

A fő értelmező ciklus

Először is megnyitjuk a naplófájlt és alapértéket adunk néhány változónak (akik RedHatet használnak, azok a /var/log/messages fájlt írják a /var/log/syslog helyére):

```
#!/usr/bin/perl
open(LOGFILE, "/var/log/syslog");
$firstdate = "";
$date = "";
$total_traffic = 0;
```

Most végiglépkedünk a naplófájl minden során:

```
while ( <LOGFILE> ) {
```

Kihagyjuk az összes bejegyzést, ami nem a tűzfalhoz tartozik:

```
next unless /Packet log/;
```

Értelmezzük a sort:

```
chomp;
@log = split;
($month,$day,$time,$policy,$proto,$ipsource,$ipdest,
     ➥$tot_len) = @log[0,1,2,8,10,11,12,13];
```

Ezután kiszámoljuk a dátumot és mentjük a napló első időpontját. Ahogy továbbhaladunk, a pillanatnyi keltezést mindig a legutolsó dátumként tároljuk, így az utolsó lépés után a *lastdate* változó a naplóban található utolsó dátumot tartalmazza majd:

```
$date = $day . " " . $month . " " . $time;
if (length($firstdate) == 0) {
  $firstdate = $date;
}
$lastdate = $date;
```

Beolvassuk a protokolltípust, a forrás IP-címet, a forráskaput, a cél IP-címet, a célkaput és a csomag hosszát:

```
$proto = substr($proto, -1);
($ips, $ports) = split ":", $ipsource;
($ipd, $portd) = split ":", $ipdest;
($flush, $packetlen) = split "=", $tot_len;
```

Tároljuk a cél IP-címet egy karakterláncban, és vessük össze a forrás IP-címmel, így az adatmegjelenítő ciklusban majd végig tudjuk nézni a forrás IP-címeket és kikereshetjük, hogy mely címekhez kapcsolódtak:

```
unless ( $sourcedest{$ips} =~ /$ipd/ ) {
  $sourcedest{$ips} =
            ➥$sourcedest{$ips} . $ipd . " ";
}
```

Megszámoljuk a forrás IP-címbejegyzéseket:

```
++$source{$ips};
```

és összegezzük a teljes forgalmat:

```
$total_traffic += $packetlen;
```

Utoljára összegezzük a kiszolgálónkénti forgalmat is:

```
$traffichost{$ips} += $packetlen;
}
```

Figyeljük meg, hogy egyáltalán nem használtuk fel az összes begyűjtött adatot (például szó sem esett a kapukról), tehát rengeteg hely akad még a bővítgetésre.

Az adatmegjelenítő ciklus

Elsőként egy mutatós honlap-fejlécet rakunk ki, amint az a *2. listában* látható. Végiglépkedünk a rendezett forrás IP-címeken, majd kiírjuk a forrás IP-címet, az arról a címről érkezett csomagok számát, illetve az arról az IP-címről származó forgalmat (bájtban mérve):

```
for (sort keys %source) {
  print "<TR><TD>$_</TD> ";
  print "<TD>$source{$_} </TD>\n";
  print "<TD>$traffichost{$_} bytes</TD>\n";
```

Mivel korábban tároltuk, ki tudjuk írni azokat a cél IP-címeket, amelyekre a pillanatnyi forrás-IP-ről csomagok továbbítódtak:

```
$tmp1 = $sourcedest{$_};
if (length($tmp1) gt 0) {
  print "<TD>\n";
  @lt1 = split " ", $tmp1;
```

2. lista Weblap-fejléc

```
print "Content-type: text/html\n\n";
print "<HTML>\n";
print "<HEAD><TITLE>ipchains Log
       ➥Scanner</TITLE></HEAD>\n";
print "<BODY bgcolor=\"#000080\"
       ➥text=\"#FFFFFF\" link=\"#BA3E8F\"
       ➥vlink=\"#BD66DE\"
       ➥alink=\"#FF0000\">\n";
print "<H1><i>inside-control</i></H1>\n";
print "<p>Log searched for network access from ";
print "<font color=\"#FFAAAA\">
       ➥$firstdate</font> to ";
print "<font color=\
       ➥"#FFAAAA\">$lastdate</font>";
print ".\n    </p>\n";
print "<H2>Total Traffic Volume:
       ➥<font color=yellow>";
print "$total_traffic bytes</font></H2>\n";
print "<hr>\n";
print "<center>\n";
print "<TABLE border=2>\n";
print "<TR><TH>Source IP</TH><TH>Packets</TH>";
print "<TH>Traffic from this host</TH>";
print "<TH>Hosts contacted</TH>";
print "</TR>\n";
```

```
  for(sort @lt1) {
    printf "$_ <br>\n";
  }
  print " </TD>\n";
}
print " </TR>\n";
}
```

Végül kiírjuk a HTML végét:

```
print "</TABLE>\n";
print "</center>\n";
print "</BODY></HTML>\n";
```

A letölthető inside-control

Az általam tényegesen megvalósított inside-control parancsállomány az itt bemutatottnál sokkal gazdagabb lehetőségekkel rendelkezik. A program a ➲ http://www.iris-tech.net/hdsl-fw címről tölthető le. Legfontosabb kiegészítő tulajdonsága: tetszőleges nevek megjelenítésére képes IP-címek helyett a *Source IP* (forrás-IP) oszlopban. Ezt egy nagyon egyszerű szöveges adatbázis segítségével tudja megvalósítani, amely az IP-számokat nevekhez rendeli. A fájl formátuma azonos a /etc/hosts fájl formátumával, így akár ezt a fájlt is használhatjuk, amennyiben az a belső hálónak megfelelően lett feltöltve. Az *IP nevek* adatbázis pontos helyét a megfelelő változó ($useripdb) átírásával adhatjuk meg a parancsfájl elején.

Egy keresőrendszert is beépítettek, amely lehetővé teszi, hogy egy adott forrás IP-címre (vagy az *IP név* adatbázisban megtalálható megfelelő névre) rákeressünk a naplóban. A kereső űrlap akkor jelenik meg, ha a CGI-t értékek nélkül hívjuk meg a böngészőből. Az értékátadás a `GET` metódussal történik.

A főciklus tartalmaz továbbá néhány adatérvényesítési eljárást (a mag nem mindig képes helyesen naplózni, különösen akkor, ha kis memória- vagy processzorteljesítmény mellett fut), és néhány kapufüggő adatot is tárol. Végezetül a parancsfájl webes csatolófelület nélkül is meghívható. Egyszerűen csak adjunk át valamilyen értéket az inside-controlnak, és a HTML-kimenet helyett hagyományos kimenetet kapunk. A keresendő forrás-IP karaktersorozatát (vagy az *IP név* adatbázisban található megfelelő nevet) a programnak a `-t` kapcsolóval adhatjuk át.

Megjegyzések és figyelmeztetések

E cikk írójának az volt a célkitűzése, hogy néhány tervezési alapelvet mutasson be és tanácsokkal szolgáljon – semmiképpen nem az, hogy a naplófigyelési gondokra előrecsomagolt megoldást szolgáltasson. Számos dolog akad, amiben az inside-control parancsállomány fejlesztésre szorul, például a teljesítmény és a biztonság területén.

A következőkben néhány szót ejtenék az inside-control jellegzetességeiről, különös tekintettel a biztonságot érintő kérdésekre.

Ahhoz, hogy a CGI a számítógép naplófájljait (/var/log/syslog vagy /var/log/messages) olvashassa, mindenki számára olvashatóvá kell tenni. Ez a `chmod +r /var/log/syslog` paranccsal érhető el. Csakhogy ez nem túl biztonságos megoldás, hiszen mindenkinek jogot ad a naplófájlok olvasgatására. Lényegesen jobb lenne rávenni a webkiszolgálót, hogy az inside-controlt egy adott csoportjogosultsággal futtassa, és ezután a naplófájlokat ehhez a csoporthoz rendelje.

A cikk elolvasása után bizonyára sokan jutnak arra a következtetésre, hogy a tűzfalnak egy webkiszolgálót is szükséges futtatnia, hiszen az inside-controlnak a tűzfal naplófájljait olvasni kell tudnia. Webkiszolgálót rakni egy tűzfalra valójában óriási biztonsági rés: eszményi esetben a tűzfal semmiféle démonszolgáltatást nem futtat és a teljes karbantartást konzolon keresztül végzik. Ha távoli felügyelet szükséges, az egyetlen szóba jöhető szolgáltatás, amely egy tűzfalra feltehető, az SSH, azaz a biztonságos héjprogram (secure shell) program.

Az inside-control futtatása még ilyenkor is megoldható a belső hálózaton egy külön webkiszolgáló felállításával, amely egyúttal a tűzfal syslog kiszolgálójaként is működik.

A tűzfal naplója rövid idő alatt könnyedén feltöltheti a teljes lemezterületet. Hogy elkerüljük tűzfalunk merevlemezének lebénulását (ami az internetkapcsolatok leállásához vezethet), a naplózandó forgalom mennyiségétől függően megfelelően nagy naplófájlhelyet kell biztosítanunk. A nagy adatmennyiségű szolgáltatások (azaz általában a HTTP-, FTP-, SMTP-, NetBIOS-, LPD- és az adatbázisszolgáltatások) esetében egy második, legalább 20 GB méretű merevlemez használatát javaslom, amelyen mindössze egyetlen, a /var/log-ra csatolt lemezterület létezzen.

Befejezésképpen: még rengeteg lehetőség nyílik fejlesztésre az egész parancsfájlban, különösen a főciklusban. Sokkal több adatot lehetne felhasználni az egyes sorokból, mint ahogy ebben a rövid példában tettük. Ennek ellenére az sem baj, ha nem mutatunk meg túl sok részletet; máskülönben az egész önműködő naplófigyelés értelmetlenné válik. Ha minden elérhető részletet megjelenítenénk, a gyanús bejegyzéseket kezelhetetlenül nagy mennyiségű forgalmi naplóban kellene megkeresnünk.

Leo Liberti
az olaszországi IrisTech cég műszaki igazgatója, amely vásárlóit webalapú alkalmazásokkal és sokféle egyéb elektronikai szolgáltatással látja el. Szabadidejét annak szenteli, hogy annyi étteremben étkezzék, amennyiben csak tud.