4. lista A daysumm parancsfájl

```
#!/bin/sh
mailto="admin@domain.com"
timestamp='date '+%A - %B %d %Y
etc="/var/spool/smtpd/etc"
date='date '+%m%d%Y
email_total='cat $etc/email-log.$date'
virus_total='cat $etc/virus-log.$date'


mail -s "DAILY E-MAIL SUMMARY" $mailto <<eoi


  Date:     $timestamp
  E-Mail total: $email_total
  Virus total:  $virus_total

eoi

/usr/sbin/sendmail -q

exit 0
```

séges, mert többé semmilyen MTA nem fut a gépünkön.
Az ***smtpd*** csomag nem MTA, hanem egy tároló- és továbbítócsomag. Úgy is elképzelhetjük, mint egy kifejezetten levéltovábbításra kihegyezett programot. E külön parancs nélkül tehát soha egyetlen levelet sem kaphatnánk meg a tűzfalról.
Itt az ideje, hogy az egész folyamatot a cron démon segítségével önműködővé tegyük. Ezt a uucp-felhasználó személyes crontab állományának felhasználásával fogjuk megtenni. Lépjünk be uucp-felhasználóként, majd adjuk ki a `crontab -e` parancsot, ami az uucp-felhasználó cron tábláját nyitja meg szerkesztéshez. A `scanmail`, `daysumm` és `sigupdate` parancsfájlok részére hozzuk létre a következő bejegyzéseket:

```
MAILTO=""

# A scanmail parancsfÆjl minden kØt percben
# fusson le
*/2 * * * * /var/spool/smtpd/bin/scanmail

# A daysumm parancsfÆjl minden nap 11:59-kor
# induljon el
59 23 * * * /var/spool/smtpd/bin/daysumm

# A sigupdate minden cs t rt k n 4:00-kor
# fusson le.
0 16  * * 4 /var/spool/smtpd/bin/sigupdate
```

Természetesen a futásidőpontokat megváltoztathatjuk úgy, hogy megfeleljenek az igényeinknek. Az, hogy a `scanmail`-t milyen sűrűn futtassuk le, főként a napi levélforgalmunktól függ. Ha a napi mennyiség tízezer levél felett van, a magam részéről az időközt két percben határoznám meg, az `smtpfwdd`-t pedig ötpercenkénti futásra állítanám be. Így nem küldünk egyszerre hatalmas levélcsomagokat a belső kiszolgálóra. Ha naponta 1000 vagy ennél kevesebb levéllel kell csak számolnunk, elég, ha a `scanmail` minden tizedik percben fut le, az `smtpfwd` pedig ötpercenként néz körül. Ne feledjük el a `MAILTO=""` kifejezést kitenni a crontab-bejegyzések elejére! Ez azért szükséges, hogy a crond a végrehajtott cron-feladatokról ne küldjön levelet az uucp-felhasználónak. Ha minden két percben egy levél érkezik, az gyorsan felgyülemlik, és az uucp-felhasználó soha nem ellenőrzi a leveleit.
Felállítottam egy Samba-megosztást is, hogy a windowsos gépemről is hozzáférhessek a ***/var/spool/smtpd*** könyvtárszerkezethez. A főként Windowst használó rendszergazdáknak ez a beállítás hasznos lehet – így nem kell mindig SSH-kapcsolatot nyitnom, valahányszor vírusra figyelmeztető levelet ellenőrzök.
A következő sorokat kell a ***/etc/smb.conf*** fájlba illeszteni:

```
[mail-gate]
       Comment = LevØltßzfal k nyvtÆrak
       Path = /var/spool/smtpd
       Valid users = nev nk
       Admin users = nev nk
       Browseable = no
       Read only = no
```

ahol a „nevünk” természetesen a Samba-felhasználói nevünket jelenti. Ami még hiányzik ahhoz, hogy tűzfalunkról az összes bejövő SMTP-kapcsolatot az új levélszűrő kiszolgálónkra irányítsuk, az, hogy tűzfalunknak IP-álcázást kell használnia. Egyszerűen adjuk ki a következő parancsot:

```
ipmasqadm portfw -a -P tcp -L tßzfal 25
➥-R cØlgØp 25
```

ahol a `tßzfal` a tűzfalunk címe, a `cØlgØp` pedig az új levélszűrő gépünk címe. Amennyiben levélszűrő tűzfalunkat közvetlenül a már meglévő tűzfalunkon szeretnénk futtatni, semmin sem kell változtatnunk.
Ha más típusú tűzfalrendszert használunk, olvassuk el a leírást, hogy megtudjuk, miképpen állíthatjuk be a kapuátirányítást. Remélhetőleg nem okoz nagy gondot, de előfordulhat, hogy kapcsolatba kell lépnünk a termék készítőjével. Ne feledjük el ezt az átirányító parancsot betenni az indító parancsfájlokba, hogy túlélje a rendszerindításokat.
Ha minden jól ment, végre működő levélvírusszűrő tűzfallal rendelkezünk. Próbáljunk meg küldeni magunknak néhány próbaüzenetet valamelyik ingyenes webes levelezőszolgáltatótól, hogy lássuk, minden jól működik-e. Én például adott időközönként küldök magamnak egy makróvírussal fertőzött állományt, hogy lássam, a rendszer még mindig helyesen működik-e. Sok dologgal lehetne még bővíteni ezt az alaprendszert. Különösen ígéretes a `daysumm` parancsfájl, amelyet jócskán fel lehetne még fejleszteni. Jelenleg épp egy CGI-parancsfájlon dolgozom, amely a pillanatnyi átlagokat – például az átlagos napi mennyiséget – az Interneten keresztül jelenítené meg. Természetesen ez csak egy út a több százból, ahogyan ez a rendszer levelezőrendszerünket megvédheti, anélkül, hogy befolyásolná a cég költségvetését. Ha valaki esetleg kitalálna valamilyen ügyes továbbfejlesztést a rendszerhez, kérem, tudassa velem. Nagyon szeretnék hallani róla.

Dave Jones
(davidashleyjones@hotmail.com)
három évig volt hálózati rendszergazda az alabamai Birminghamben. Amikor éppen nem a számítógép előtt ül, egy-egy szál jófajta dohányt szív el, vagy a feleségével és a lányával X-aktákat néz a tévében.

Gyanús adatforgalom felderítése

Használjunk psad-t IP Chains, illetve IP Tables szabálykészletünkhöz, hogy felfedezhessük a TCP- és UDP-kapuvizsgálatokat, és más hálózati gonoszkodásokat.

Alig fél éve végre kiadásra került Linux 2.4.0 rendszermag által a GNU/Linux hatalmas lépést tett a vállalati operációs rendszerek világa felé. Több magrész is fejlődött ugyan a 2.2.x sorozat óta, de egyik sem olyan jelentős mértékben, mint a tűzfalkód. A 2.4.x rendszermagsorozatban megjelent a Netfilter (lásd a ***Linuxvilág*** 2001. októberi számát, 27–31. oldal), amely a 2.2.x sorozat régi IP Chains tűzfalkódjának helyére lépett, és több olyan képességgel is rendelkezik, amelyre egy valódi kereskedelmi tűzfalnak szüksége lehet. Alaposságát képességei bizonyítják a leginkább: a DoS-védelem (DoS = Denial of Service, szolgáltatásmegtagadás alapú támadás, azaz tömeges kérelemmel való bombázás a kiszolgáló lebénítása céljából), illetve sűrűségkorlátozás, hálózati címátalakítás (NAT), MAC-címszűrés, és végezetül, de nem utolsó sorban, tetszés szerinti TCP-jelzőkombináción alapuló TCP-csomagszűrés és naplózás. Ezzel szemben, az IP Chainsnek szánalmas korlátja is akad (nem végez alapos vizsgálatot), ugyanis mindössze két fajta TCP-csomagot képes megkülönböztetni aszerint, hogy a csomag SYN jelzője be van-e állítva vagy sem. A Netfilter azon képessége, hogy bármilyen tetszőleges TCP jelzőkombinációt megenged, lehetővé teszi, hogy azokat a kifinomult kapuvizsgálatokat is felderíthessük, amelyeket az Nmap segítségével bárki könnyedén a gépre engedhet. A kapuvizsgálatok működésének bemutatásához és felderítésük módozatainak az ismertetéséhez először némi Nmap-háttérismeretre (lásd még a ***Linuxvilág*** 2001. májusi számát, 45–49. oldal) lesz szükségünk.

Nmap

Az Nmap a világ legismertebb, többfajta fejlett módszert egyesítő programja, amellyel a nyitott kapukat a célgépen, illetve a hálózaton egyaránt felderíthetjük. Az Nmap-rendszer a nyitott kapuk minél tökéletesebb meghatározásához kínál többek között ujjlenyomat-elemzést és TCP-sorozatszám előrejelzést (TCP sequence number prediction) is. Az Nmap által alkalmazott három legérdekesebb TCP-vizsgálati mód a FIN-, a NULL- és a XMAS-vizsgálat. A megszokott TCP-forgalomban a FIN-csomagokat (az olyan csomagokat, amelyeknél a FIN jelző be lett állítva) a TCP-kapcsolat bármely végéről küldeni lehet, jelezvén, hogy az üzenetváltásnak vége, pontosabban nincs több küldendő adat. A FIN-vizsgálat azon az elven alapul, hogyha egy „árva” FIN-csomagot (olyan FIN-csomagot, amely semmilyen létező TCP-üzenetváltásnak nem része) küldünk egy nyitott TCP-kapura, nem kapunk visszajelzést. Ha viszont egy ilyen csomagot egy zárt kapura küldünk, a hagyományos szűretlen TCP-verem várhatóan egy RST-csomaggal válaszol. Így aztán a FIN-csomagvizsgálatnál az Nmap egyszerűen minden egyes célkapura elküld egy magányos FIN-csomagot és várja, vajon visszaérkezik-e valahonnan RST-csomag. Azok a kapuk, amelyek nem válaszoltak RST-csomaggal, nyitva vannak (vagy tűzfallal szűrtek). A NULL- és a XMAS-vizsgálat is hasonló módszeren alapul, de ahelyett, hogy csak a FIN-jelzőt állítaná be, az XMAS-vizsgálat az URG- és PSH-jelzőket is beállítja, a NULL-vizsgálat pedig olyan csomagokat készít, amelyeknek egyetlen jelzőjük sincs beállítva.

A Netfilter beállítása

A biztonságos tűzfal készítésének alapgondolata az alapértelmezett tagadó hozzáállás. Eszerint, amely adatforgalom nincs kifejezetten engedélyezve, azt a tűzfalnak meg kell tagadnia vagy el kell utasítania.
Tűzfalak esetében három kifejezést használunk, amelyek ugyan rokon értelműek, de a tűzfalak esetében más és mást jelentenek:
REJECT – elutasít (ilyenkor a tűzfal egy „elutasítva” üzenetet küld vissza),
DENY – megtagad (a tűzfal kidobja a csomagot, nincs válasz),
DROP – elvet, eldob (a tűzfal kidobja a csomagot, nincs válasz).
Továbbá a tűzfalat úgy kell beállítani, hogy minden jogosulatlan csomagot egy naplófájlba naplózzon, így azt később a rendszergazda vagy a `psad`-hoz hasonlóan önműködő naplófájlfigyelő rendszer elemezheti.
Ha valakit bővebben érdekel, hogyan tudná az IP Tablest a rendszerén beindítani, melegen ajánlom ***Rusty Russell*** Netfilter HOWTO-ját a ➲ http://netfilter.samba.org-on.

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot source   destination
ACCEPT     all  anywhere anywhere state
           RELATED,ESTABLISHED
ACCEPT     TCP  anywhere anywhere TCP dpt:ssh
           flags:FIN,SYN,RST,PSH,ACK,URG/SYN
ACCEPT     TCP  anywhere anywhere TCP dpt:www
           flags:FIN, YN,RST,PSH,ACK,URG/SYN
LOG        TCP  anywhere anywhere LOG level
           warning prefix 'DENY ·
DROP       TCP  anywhere anywhere
```

Kapuvizsgálatok megállítása

Vajon megállíthatja-e egy ilyen IP Tables-szabályzat a kapuvizsgálatokat? Először minden TCP-csomagot (a TCP-jelzőtől függetlenül), amely a 80-as vagy a 22-es kaputól eltérő számú kapura érkezne, a négyes számú szabály szerint naplózzunk, majd az ötös számú szerint dobjuk el. Ez máris minden zajos vizsgálatnak ellátja a baját, amely a létező 65 535 TCP-kapu közül nem pont a 80-as vagy a 22-es kaput célozza. No de mi történjen azokkal a vizsgálatokkal, amelyek a 80-as vagy a 22-es kaput célozzák? Ez a vizsgálat típusától függ. A hármas és négyes szabályok elfogadják azokat a csomagokat, amelyeknek be van állítva a SYN-jelzőjük, de csak akkor, ha minden más jelző törölve van, így azok a vizsgálatok, amelyek kizárólag SYN-csomagokat használnak, sikeresek lesznek. Ugyanígy minden vizsgálat, amely rendes `TCP connect()` rendszerhívást használ, ahogyan azt minden rendes böngésző vagy SSH-ügyfél is tenné, szintén sikeres lesz, hiszen az egyes számú szabály engedélyezi a három TCP-kézfogás végrehajtását.
Az Nmap mindkét vizsgálati módszert támogatja a `-sS` (félig

1. lista. Tűzfalüzenetek beolvasása kmsgsd-vel

```
open LOG, ">> /var/log/psad/fwdata" or die
"$!\n";

while (1) {
  open FIFO, "< /var/log/psadfifo" or die "$!\n";
  $service = <FIFO>; # ne lass tsuk a chomp-pal
  if (($service =~ /Packet\slog/ || $service =~
/IN.+?OUT.+?MAC/) && $service =~
/DROP|REJECT|DENY/) {
    # a tiltott/eldobott csomag napl zÆsa
    # a fwdata fÆjlban
    my $old_fh = select LOG;
    $| = 1;  # a LOG-ba helyezz k a pufferelt
             # kimenetet
      print "$service";
      select $old_fh;
   }
}
```

3. lista Webkiszolgálóhoz csatlakozó TCP-folyamat IP Tables-üzenetei

```
Aug  4 12:16:56 myserver kernel: IN=eth1 OUT=
MAC=00:a0:cc:e2:1f:f2:00:20:78:1c:60:58:08:00
SRC=192.168.10.10 DST=10.10.10.50 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=25415 DF PROTO=TCP SPT=2591
DPT=80 WINDOW=32120 RES=0x00 SYN URGP=0

Aug  4 12:16:56 myserver kernel: IN= OUT=eth1
SRC=10.10.10.50 DST=192.168.10.10 LEN=60 TOS=0x00
PREC=0x00 TTL=64 ID=0 DF PROTO=TCP SPT=80
DPT=2591 WINDOW=5792 RES=0x00 ACK SYN URGP=0

Aug  4 12:16:56 myserver kernel: IN=eth1 OUT=
MAC=00:a0:cc:e2:1f:f2:00:20:78:1c:60:58:08:00
SRC=192.168.10.10 DST=10.10.10.50 LEN=52 TOS=0x00
PREC=0x00 TTL=64 ID=25416 DF PROTO=TCP
SPT=2591 DPT=80 WINDOW=32120 RES=0x00 ACK URGP=0
```

nyílt vagy SYN-vizsgálat) és a `-sT` (TCP `connect()` vizsgálat) parancssori kapcsolókkal. Bármely más vizsgálati módszert, amely nem az ezekre a kapukra irányuló szabályos TCP-forgalmon alapul, megakadályozunk és naplózunk. Ide értendő a FIN-, XMAS- és NULL-vizsgálat, amit a korábbi Nmap-részben említettünk, illetve a SYN/FIN- és az ACK-vizsgálat is.
Most ugyan végre bizonyosak lehetünk abban, hogy az IP Tables tűzfalunk képes megállítani a kapuvizsgálatokat, de ne üldögéljünk boldogan a babérjainkon! Egyáltalán nem elég csak megállítani a vizsgálatokat, a lehető legmegbízhatóbban azonosítanunk is kell őket, hiszen a kapuvizsgálat gyakran egy sokkal komolyabb támadásnak lehet az előjele.

A psad bemutatása

A Port Scan Attack Detector (`psad`) Perl nyelven íródott program, amelyet arra terveztek, hogy a szigorú IP Chains, illetve IP Tables szabálykészletek használatával TCP- és UDP-kapuvizsgálatokat derítsen fel. Lehetőségünk nyílik néhány előre beállítható veszélyességi szint (természetesen rendelkezésre áll néhány célravezető beépített szint is) létrehozására, kérhetünk figyelmeztetést levélben is, igény szerint önműködően letilthatjuk a támadó IP-címet az IP Chains, illetve IP Tables tűzfalszabályok dinamikus átállításával. Ezenkívül bőbeszédű riasztásokat kaphatunk, amelyek tartalmazzák a forrást, a célt, a vizsgálat alá vont kaputartományt, a kezdés és a befejezés időpontját, és a `dns` és `whois`-keresések eredményét. IP Tables tűzfal használata esetén a `psad` képes kihasználni a továbbfejlesztett naplózási képességet, és fel tudja deríteni az olyan különösen gyanús vizsgálatokat, mint amilyen a FIN, XMAS vagy a NULL, azaz amelyeket valaki az Nmap segítségével könnyen a gép ellen fordíthat. Továbbá a `psad` több, a ***Snort*** behatolásfelderítő rendszerhez tartozó TCP- és UDP-aláírást tartalmaz, amelyek alapján fényt deríthet a vizsgálatokra, illetve a különféle hátsó ajtókra (backdoor) – például az EvilFTP, GirlFriend, SubSeven – és a DdoS-eszközök (mstream, shaft) adatforgalmára. A `psad` a GNU Public License alá tartozó ingyenes program és a ➲ http://www.cipherdyne.com honlapról tölthető le.
A `psad` alapfeladata, hogy a segítségével értelmezzük az IP Chains vagy IP Tables tűzfal által készített naplóüzeneteket, és felderíthessük a gyanús hálózati forgalmat. A feladat kivitelezéséhez a `psad`-nak hatékony módszerre van szüksége, amivel a tűzfal által a rendszernaplóba írt üzenetekből a számára fontos adatokat kigyűjtheti. A `psad` ezért telepítéskor egy `psadfifo` nevű nevesített csővezetéket (named pipe) készít a */var/log/* könyvtárban, és beállítja a rendszernaplódémont (`syslogd`), hogy a ***kern.info*** üzeneteket ebbe a vezetékbe írja. A syslog nyelvezetében a ***kern*** szolgáltatás által jelentett IP Chains és IP Tables naplóüzenetek ***info*** naplószinten jelennek meg. A `psad` által végzett feladatok oroszlánrészét két külön démon, a `kmsgsd` és a `psad` végzi.

kmsgsd

A `kmsgsd` démon viszonylag egyszerű felépítésű. Feladata mindössze annyiból áll, hogy megnyitja a `psadfifo` vezetéket, majd minden olyan ***kern.info*** üzenetet beolvas, amely arra utalhat, hogy az IP Chains, illetve IP Tables egy csomagot elvetett vagy elutasított, végül minden ilyen üzenetet a */var/log/psad/fwdata* `psad` adatfájlba ír. Mivel a `kmsgsd`-be épített `regex` (szabványos kifejezés-kereső) csak az olyan csomagokat keresi ki, amelyeket elvetettek vagy elutasítottak, az ***fwdata*** fájl már megszűrt adatfolyam lesz, amely csak olyan adatot tartalmaz, ami biztonsági szempontból fontos lehet. Ez az adathalmaz azonban csak annyira lehet teljes és beszédes, amennyire a tűzfal naplózza őket, ezért van tehát szükség a lehető legszigorúbb tűzfalszabályokra.
Az IP Tables nem támogatja a naplózás lehetőségét semmilyen szabályhoz, amely csomagokat vet vagy utasít el. Ezt a gondot azonban könnyen áthidalhatjuk, ha az elvetést végző szabály elé egy `--log-prefix` kapcsolót tartalmazó naplózási szabályt teszünk. Ezt figyelhetjük meg a `simplefirewall.sh` negyedik szabályában. Az *1. listában* látható `kmsgsd`-kódrészlet azt mutatja be, hogyan olvas a `psadfifo` tűzfalüzeneteket nevesített csővezetékről, és miképpen lehet szabványos kifejezések segítségével az IP Chains vagy az IP Tables által elvetett csomagok üzeneteit kiszűrni.